PPROVED BY DRAFTSMAN	O.G. FIG.	
	CLASS	SUBCLASS

5920836

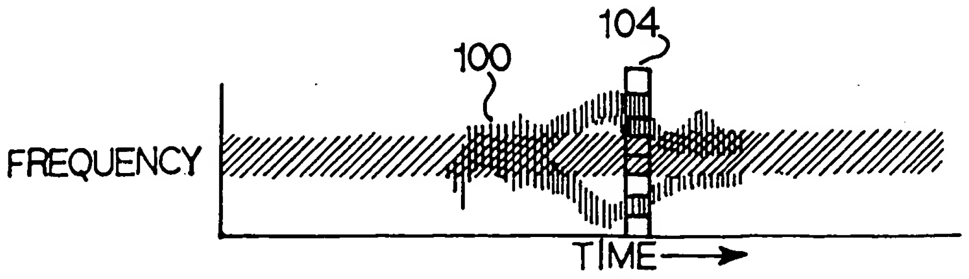


FIG.1
PRIOR ART

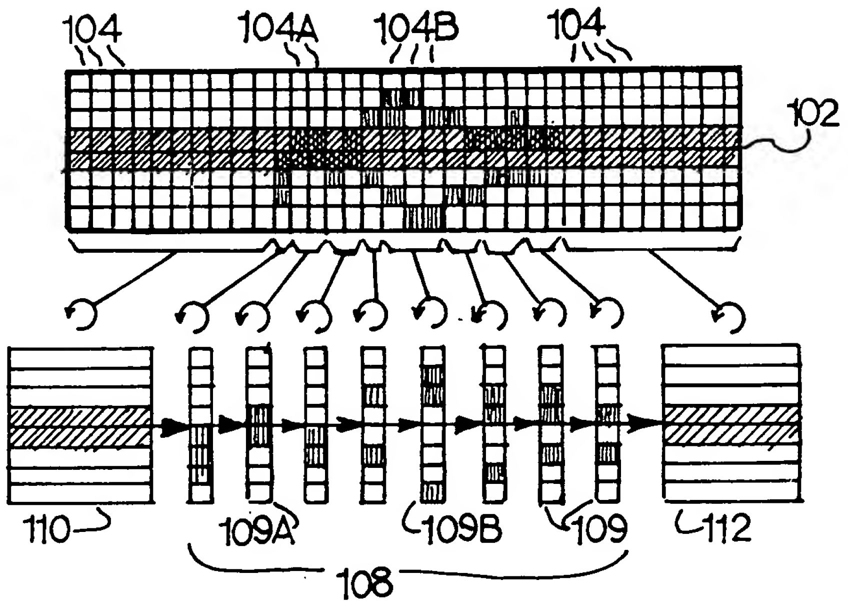


FIG.2
PRIOR ART

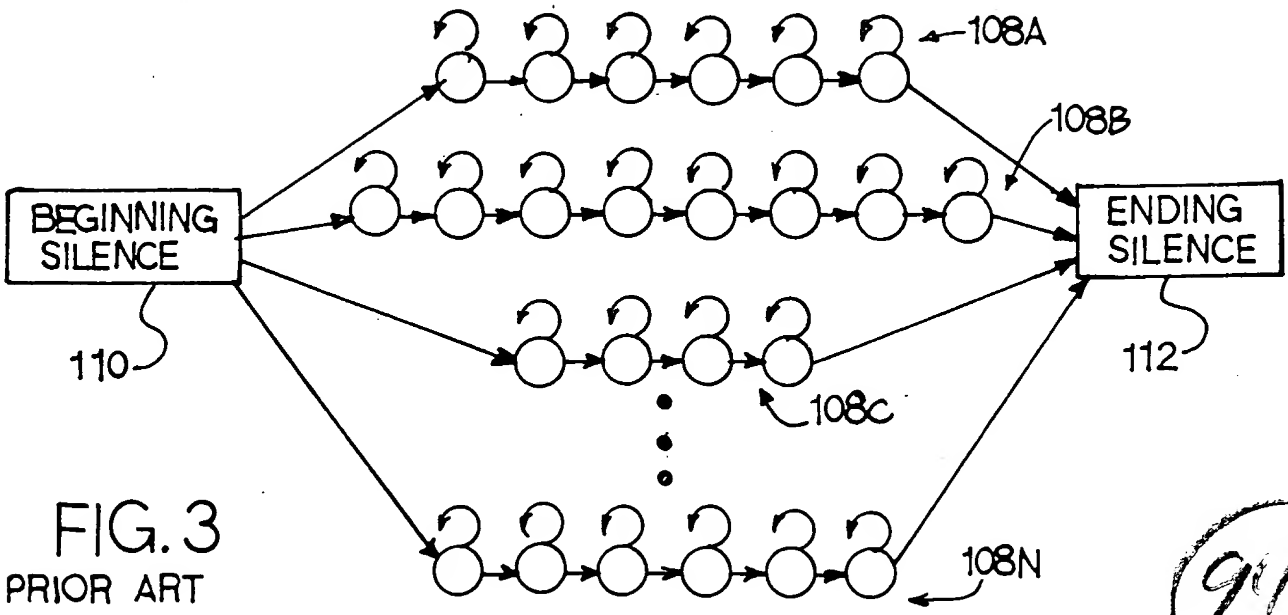
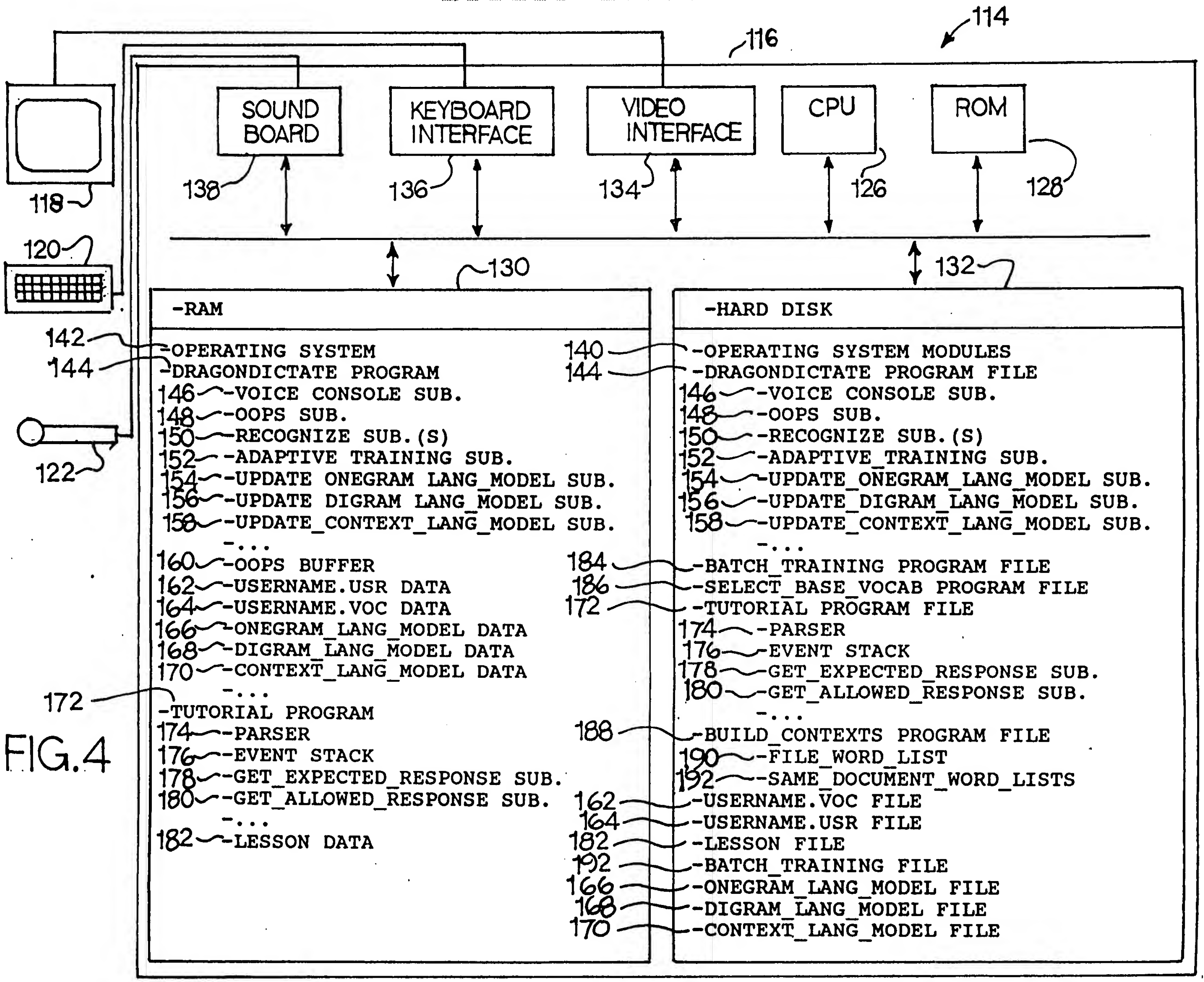


FIG.3
PRIOR ART

08828813.062692



114
116
SOUND BOARD
KEYBOARD INTERFACE
VIDEO INTERFACE
CPU
ROM
138
136
134
126
128
118
120
122
130
132
-RAM
142 -OPERATING SYSTEM
144 -DRAGONDICTATE PROGRAM
146 -VOICE CONSOLE SUB.
148 -OOPS SUB.
150 -RECOGNIZE SUB.(S)
152 -ADAPTIVE TRAINING SUB.
154 -UPDATE ONEGRAM LANG_MODEL SUB.
156 -UPDATE DIGRAM LANG_MODEL SUB.
158 -UPDATE_CONTEXT_LANG_MODEL SUB.
-...
160 -OOPS BUFFER
162 -USERNAME.USR DATA
164 -USERNAME.VOC DATA
166 -ONEGRAM_LANG_MODEL DATA
168 -DIGRAM_LANG_MODEL DATA
170 -CONTEXT_LANG_MODEL DATA
-...
172 -TUTORIAL PROGRAM
174 -PARSER
176 -EVENT STACK
178 -GET_EXPECTED_RESPONSE SUB.
180 -GET_ALLOWED_RESPONSE SUB.
-...
182 -LESSON DATA
-HARD DISK
140 -OPERATING SYSTEM MODULES
144 -DRAGONDICTATE PROGRAM FILE
146 -VOICE CONSOLE SUB.
148 -OOPS SUB.
150 -RECOGNIZE SUB.(S)
152 -ADAPTIVE_TRAINING SUB.
154 -UPDATE_ONEGRAM_LANG_MODEL SUB.
156 -UPDATE_DIGRAM_LANG_MODEL SUB.
158 -UPDATE_CONTEXT_LANG_MODEL SUB.
-...
184 -BATCH_TRAINING PROGRAM FILE
186 -SELECT_BASE_VOCAB PROGRAM FILE
172 -TUTORIAL PROGRAM FILE
174 -PARSER
176 -EVENT STACK
178 -GET_EXPECTED_RESPONSE SUB.
180 -GET_ALLOWED_RESPONSE SUB.
-...
188 -BUILD_CONTEXTS PROGRAM FILE
190 -FILE_WORD_LIST
192 -SAME_DOCUMENT_WORD_LISTS
162 -USERNAME.VOC FILE
164 -USERNAME.USR FILE
182 -LESSON FILE
192 -BATCH_TRAINING FILE
166 -ONEGRAM_LANG_MODEL FILE
168 -DIGRAM_LANG_MODEL FILE
170 -CONTEXT_LANG_MODEL FILE

FIG.4

```
-DRAGONDICTATE PROGRAM~144
     -INITIALIZE~204
     -TERMINATE AND STAY RESIDENT~206
     -GET USER INPUT BY MONITORING KEYSTROKE INTERRUPTS AND, IF
     MICROPHONE IS ON, UTTERANCE INTERRUPTS~208
     -IF RECEIVED KEYSTROKE IS:~210
          -"+", CALL VOICE CONSOLE SUBROUTINE
          -"-", CALL OOPS BUFFER SUBROUTINE
          -...
          -ANY OTHER KEY, PASS TO ACTIVE PROGRAM
     -IF RECEIVE UTTERANCE~212
          -CALL RECOGNIZER~214
          -IF BEST SCORING WORD IS:~216
               -CHOICE COMMAND SELECTING A WORD IN ALTERNATE CHOICE
               WINDOW~226
                    -IF CHOICE COMMAND SELECTS OTHER THAN BEST SCORING
                    WORD~228
                         -SIMULATE TYPING NUMBER OF BACKSPACE CHARACTERS
                         EQUAL TO NUMBER OF CHARACTERS IN FIRST CHOICE
                         WORD~230
                         -SIMULATE TYPING CHARACTERS OF SELECTED WORD~232
                    -REMOVE CHOICE WINDOW~234
                    -MAKE SELECTED WORD FIRST CHOICE~236
                    -SET UTTERANCE'S CONFIRMED_FLAG~254
                    -CALL ADAPTIVE_TRAINING SUBROUTINE FOR CONFIRMED
                    UTTERANCE AND FIRST CHOICE WORD~256
               -"CHOOSE-10", OR "SCRATCH THAT"~360
                    -BACKSPACE NUMBER OF CHARACTERS IN BEST SCORING
                    WORD~362
                    -REMOVE CHOICE WINDOW~364
                    -REMOVE UTTERANCE'S ENTRY IN OOPS BUFFER~366
               -"OOPS"~368
                    -CALL OOPS_SUBROUTINE~370
               -...
               -NOT ONE OF ABOVE COMMANDS~218
                    -REMOVE PREVIOUS CHOICE WINDOW IF ANY~223
                    -SIMULATE TYPING OF UTTERANCE'S BEST SCORING WORD~220
                    -PLACE CHOICE WINDOW ON SCREEN NEAR CURSOR~222
                    -IF CONFIRMED_TRAINING_ONLY_FLAG IS FALSE OR IF THE
                    CONFIRMED_FLAG OF THE OLDEST ENTRY IN THE OOPS BUFFER
                    IS SET~392
                         -CALL ADAPTIVE TRAINING SUBROUTINE FOR TOKEN OF
                         THE OLDEST ENTRY IN THE OOPS BUFFER AGAINST THAT
                         ENTRY'S FIRST CHOICE WORD, UNLESS ALREADY
                         DONE~394
```

FIG. 5

```
                    -CALL UPDATE ONEGRAM, UPDATE DIGRAM,
                    AND UPDATE CONTEXT LANG MODEL
                    SUBROUTINES BASED ON OLDEST ENTRY'S
                    FIRST CHOICE WORD~396
                -IF SAVING_TOKEN_FLAG IS SET, SAVE OLDEST
                ENTRY'S TOKEN LABELED WITH ITS FIRST CHOICE WORD
                IN A FILE, BUFFERING SAVES TO REDUCE DISK
                ACCESS~398
            -ADD NEW ENTRY TO OOPS BUFFER FOR LAST UTTERANCE,
            INCLUDING ITS TOKEN, NINE BEST SCORING WORDS, AND A
            ZEROED CONFIRM_FLAG~400
```

FIG. 5 CONT.

```
-VOICE CONSOLE SUBROUTINE~146
    -IF SYSTEM HAS ONE OR MORE USER FILES DEFINED~402
        -ENABLE FULL VOICE CONSOLE MENU
    -IF NOT~404
        -LIMITED VOICE CONSOLE MENU TO LOAD USER OR EXIT
    -VOICE CONSOLE LOOP~406
        -CLEAR OTHER PROMPTS, IF ANY, AND DISPLAY VOICE CONSOLE
        MENU~408
        -GET USER INPUT~410
        -IF INPUT IS:~412
            -"LOAD USER"~414
                -PROMPT FOR USER NAME~416
                -GET INPUT~420
                -IF USER ENTERS A NEW USER NAME~422
                    -PROMPT IF WANT TO CREATE NEW USER~424
                    -IF NOT, RETURN TO TOP OF VOICE CONSOLE LOOP~426
                    -IF SO~428
                        -PROMPT IF WANT TO RUN TUTORIAL~430
                        -IF USER SELECTS YES~432
                            -EXIT VOICE CONSOLE
                            -LOAD AND RUN TUTORIAL
                        -ELSE~434
                            -EXIT VOICE CONSOLE
                            -LOAD AND RUN SELECT_BASE_VOCAB
                            PROGRAM
                -SELECT USER'S .VOC AND .USR FILES FOR USE BY
                RECOGNIZER~446
                -EXIT VOICE CONSOLE~448
            -...
            -"UTILITIES"~450
                -DISPLAY UTILITIES MENU~452
```

FIG. 6

```
                    -GET INPUT~452
        -IF INPUT IS:
        -...
             -"PARAMETERS",~454
                  -DISPLAY PARAMETERS MENU~456
                  -GET INPUT~456
                  -IF INPUT IS
                       -...
                       -"CONFIRMED TRAINING ONLY", SET
                       CONFIRMED_ TRAINING_ONLY_FLAG~468
                       -"SAVE TOKEN", SET SAVE_TOKEN_FLAG~460
                       -...
                  -...
        -...
   -...
```

FIG. 6 CONT.

```
-OOPS SUBROUTINE~148
     -MAKE 2ND MOST RECENT UTTERANCE IN OOPS BUFFER THE CURRENT OOPS
     WORD~372
     -REPEAT UNTIL EXIT FROM WITHIN~374
          -DISPLAY OOPS MENU WITH ONLY CURRENT OOPS WORD HAVING ALTERNATE
          CHOICES SHOWN~376
          -GET INPUT~378
          -IF INPUT IS:~380
               -CHOOSE-1 OR OKAY, REMOVE OOPS MENUS, MAKE ALL CORRECTIONS
               TO OUTPUT, AND EXIT OOPS SUBROUTINE~381
               -CHOOSE-2, SELECT SECOND CHOICE WORD, REMOVE OOPS MENUS,
               MAKE ALL CORRECTIONS TO OUTPUT, AND EXIT OOPS
               SUBROUTINE~382
               -...~386
               -SELECT-1, REMOVE ALTERNATE CHOICE MENU FROM CURRENT OOPS
               WORD~383
               -SELECT-2, REMOVE ALTERNATE CHOICE MENU FROM CURRENT OOPS
               WORD, MAKE SECOND CHOICE WORD THE FIRST CHOICE~384
               -...~386
               -LEFT-1, MAKE WORD ONE LEFT OF CURRENT OOPS WORD THE
               CURRENT OOPS WORD~388
               -LEFT-2, MAKE WORD TWO LEFT OF CURRENT OOPS WORD THE
               CURRENT OOPS WORD~390
               -...~386
               -RIGHT-1, MAKE WORD ONE RIGHT OF CURRENT OOPS WORD THE
               CURRENT OOPS WORD~394
               -...~386
```

FIG. 7

```
-OOPS BUFFER~160
     -ENTRY1
     -ENTRY2
     -ENTRY3~238
     -ENTRY4
     -ENTRY5~238
     -ENTRY6~238
     -ENTRY7
     -ENTRY8
     -ENTRY9
     -ENTRY10
     -ENTRY11
     -ENTRY12
     -READ/WRITE POINTER~240
```

FIG. 8

```
-OOPS BUFFER ENTRY~238
     -TOKEN~244
     -WORD_1~246A
     -WORD_2
     -WORD_3~246
     -WORD_4
     -WORD_5~246
     -WORD_6~246
     -WORD_7
     -WORD_8
     -WORD_9
     -VOCABULARY~248
     -STATE~250
     -CONFIRMED_FLAG~252
```

FIG. 9

```
-USERNAME.VOC FILE~162
     -LIST OF WORDS~260
          -FOR EACH
               -WORD~263
               -PHONEME SPELLING LIST~262
                    -PHONETIC SPELLINGS~263
               -PREFILTERING WORD START~264
     -LIST OF STATES~266
          -FOR EACH
               -STATE~267
               -LIST OF WORDS OR INCLUDED STATES~268
                    -FOR EACH
                         -WORD OR STATE~269
```

FIG. 10

```
                    -TRANSITION TO ANOTHER STATE~270
          -EXTRA DATA (SUCH AS KEYSTROKE SEQUENCE)~272
     -DEFAULT TRANSITION~274
     -DEFAULT EXTRA DATA~276
```

FIG. 10 CONT.

```
-USERNAME.USR FILE~164
     -PREFILTERING MODELS~280
     -PIC TABLE~282
          -FOR EACH PHONEME TRIPLE
               -ITS ASSOCIATED SEQUENCE OF PELS~284
               -DURATION MODEL~286
     -PEL MODEL LIST~288
          -FOR EACH PEL
               -PEL ID~291
               -1 AMPLITUDE PARAMETER~290
               -7 SPECTRAL PARAMETERS~292
               -12 CEPSTRAL PARAMETERS~294
     -HELPER MODEL LIST~296
          -FOR EACH WORD FOR WHICH USER UTTERANCES SCORE POORLY AGAINST
          PHONETIC MODEL
               -WORD~298
               -PHONETIC MODEL OF WORD, IF ANY~300
               -SEQUENCE OF PELS~302
               -PREFILTERING WORD START~303
```

FIG. 11

```
-ADAPTIVE_TRAINING SUBROUTINE~152
     -ADJUST WEIGHT TO BE GIVEN TOKEN IN TRAINING ACCORDING TO SUCH
     FACTORS AS STATE OF CONFIRMED_FLAG~304
     -CALL WORD_TRAINING FOR WORD, TOKEN, AND WEIGHT~306
```

FIG. 12

```
-TRAINING SUBROUTINE (TOKEN LIST, WORD MODEL)~326
     -FOR EACH TOKEN IN TOKEN LIST~330
          -TIME ALIGN AND SCORE PARAMETER VECTORS OF TOKEN AGAINST PELS
          OF WORD MODEL~332
     -UPDATE PELS OF WORD MODEL WITH VECTORS TIME ALIGNED AGAINST
     THEM~334
```

FIG. 13

```
-TRAIN_NEW_MODEL SUBROUTINE (TOKEN LIST)~336
     -SET PEL_NUMBER IN PROPORTION TO AVERAGE LENGTH OF TOKENS IN TOKEN
     LIST~338
     -DIVIDE EACH TOKEN INTO PEL_NUMBER SEGMENTS OF APPROXIMATELY EQUAL
     LENGTH~340
     -MAKE AN INITIAL MODEL FOR THE WORD WITH A PEL FOR EACH OF THE
     PEL_NUMBER SEGMENTS MADE IN THE TOKENS, WITH EACH PEL'S PARAMETERS
     BEING BASED ON THE VECTORS OF THE ONE OR MORE TOKENS IN ITS
     ASSOCIATED SEGMENT~342
     -REPEAT UNTIL ITERATION IMPROVES SCORE OF MATCHES BY LESS THAN
     SPECIFIED AMOUNT~344
          -FOR EACH TOKEN IN TOKEN LIST~346
               -TIME ALIGN AND SCORE PARAMETER VECTORS OF TOKEN AGAINST
               PELS OF WORD MODEL~348
          -UPDATE PELS OF WORD MODEL~350
```

FIG. 14

```
-BATCH_TRAINING PROGRAM~184
     -FOR EACH WORD FOR WHICH HAVE TOKENS~464
          -CALL WORD_TRAINING FOR THE WORD AND ITS TOKEN~466
```

FIG. 15

```
-SELECT_BASE_VOCAB PROGRAM~186
     -DISPLAY SENTENCE AND PROMPT USER TO SEPARATELY SPEAK EACH HILITED
     WORD IN THAT SENTENCE~436
     -FOR EACH WORD IN SENTENCE, STARTING WITH FIRST~438
          -HILITE WORD
          -GET NEXT UTTERANCE
          -LABEL UTTERANCE'S TOKEN AS BEING FOR HILTITED WORD
     -SCORE EACH UTTERANCE'S TOKEN AGAINST ITS LABELED WORD IN EACH OF
     BASE VOCABULARIES~440
     -ADD SCORES OF ALL UTTERANCES FOR EACH VOCABULARY~442
     -SELECT BASE VOCABULARY WITH BEST SCORE, BASING USER'S .VOC AND .USR
     FILES ON SELECTED BASE VOCABULARY~444
```

FIG. 16

```
-TUTORIAL PROGRAM~172
     -INITIALIZE~460
     -REPEAT UNTIL EXIT FROM WITHIN~461
          -GET NEXT LINE OF LESSON FILE~462
          -INTERPRET AND EXECUTE THAT LINE~463
```

FIG. 17

```
-LESSON FILE~182
     -CHAPTER1--BASE FILE SELECTION~464A
          -SET DEFAULTS FOR CHAPTER~475
          -LESSION~468A
               -DISPLAY INTRODUCTORY SCREEN
               -GET INPUT
          -...
          -SELECT BASE FILE LESSON~468B
               -RUN SELECT_BASE_VOCAB
          -...
     -CHAPTER2--INTRODUCTION TO TUTORIAL~464
     -CHAPTER3--HOW DRAGONDICTATE WORDS~464
     -CHAPTER4--THE VOICE CONSOLE AND DISABLING THE MICROPHONE~464
     -CHAPTER5--LEARNING TO DICTATE
     -CHAPTER6--BASIC PUNCTUATION
     -CHAPTER7--CORRECTING DICTATION WITH THE CHOICE LIST~464B
     -CHAPTER8--DELETING UTTERANCES WITH [CHOOSE 10]
     -CHAPTER9--SPELLING WORDS NOT ON CHOICE LIST
     -CHAPTER10-THE DICTIONARY AND ADDING NEW WORDS
     -CHAPTER11-CORRECTING OLD ERRORS WITH THE OOPS BUFFER
     -CHAPTER12-DICTATING DATES, NUMBERS, AND ADDRESSES
     -CHAPTER13-SAVING YOUR VOCABULARY FILES
     -...
     -CHAPTERN~464C
          -SET DEFAULTS FOR CHAPTER
          -BATCH TRAINING LESSON~468C
               -PROMPT USER IF WANTS TO PERFORM BATCH TRAINING~486
               -IF USER SAYS YES, CALL BATCH_TRAINING~488
               -ELSE, CONTINUE TO NEXT LESSION
               -...
          -EXIT LESSON~468D
               -PROMPT USER IF WANTS TO EXIT TUTORIAL~490
               -IF USER SAYS YES, EXIT TUTORIAL~492
               -ELSE, PROMPT USER TO CALL TUTOR MENU FOR OPTIONS~494
               -...
     -...
     -DICTATION MODULE~466A
     -GLOBAL MODULE~466B
     -TUTOR MENU MODULE~466C
          -SET DEFAULTS FOR MODULE
          -DISPLAY TUTOR MENU
          -GET IMPUT
          -BRANCH BASEDD ON INPUT
     -...
```

FIG. 18

```
-CHAPTER~464
    -SET DEFAULTS FOR CHAPTER~469
    -LESSON~468
        -OPTIONALLY DISPLAY MESSAGE~470A
        -OPTIONALLY FAKE DICTATION ACTION~470B
        -OPTIONALLY ADD ENTRIES TO STACK~470C
        -GET INPUT~470D
        -CONTINUE OR BRANCH BASED ON INPUT~470E
    -LESSON~468
    -LESSON~468
    -...
```

FIG. 19

```
-GET_EXPECTED_RESPONSE SUBROUTINE~178
    -CALL GET_ALLOWED_RESPONSE SUBROUTINE~520
    -IF RETURNS EXPECTED WORD AS USER RESPONSE~522
        -RETURN
    -IF RETURNS OTHER ALLOWED RESPONSE IN EVENT STACK~524
        -EXECUTE FUNCTION FOLLOWING THAT ALLOWED RESPONSE IN EVENT
        STACK
    -IF FUNCTION CALLED FROM EVENT STACK RETURNS WITH A "REPEAT", JUMP
    TO START OF THIS SUBROUTINE~525
```

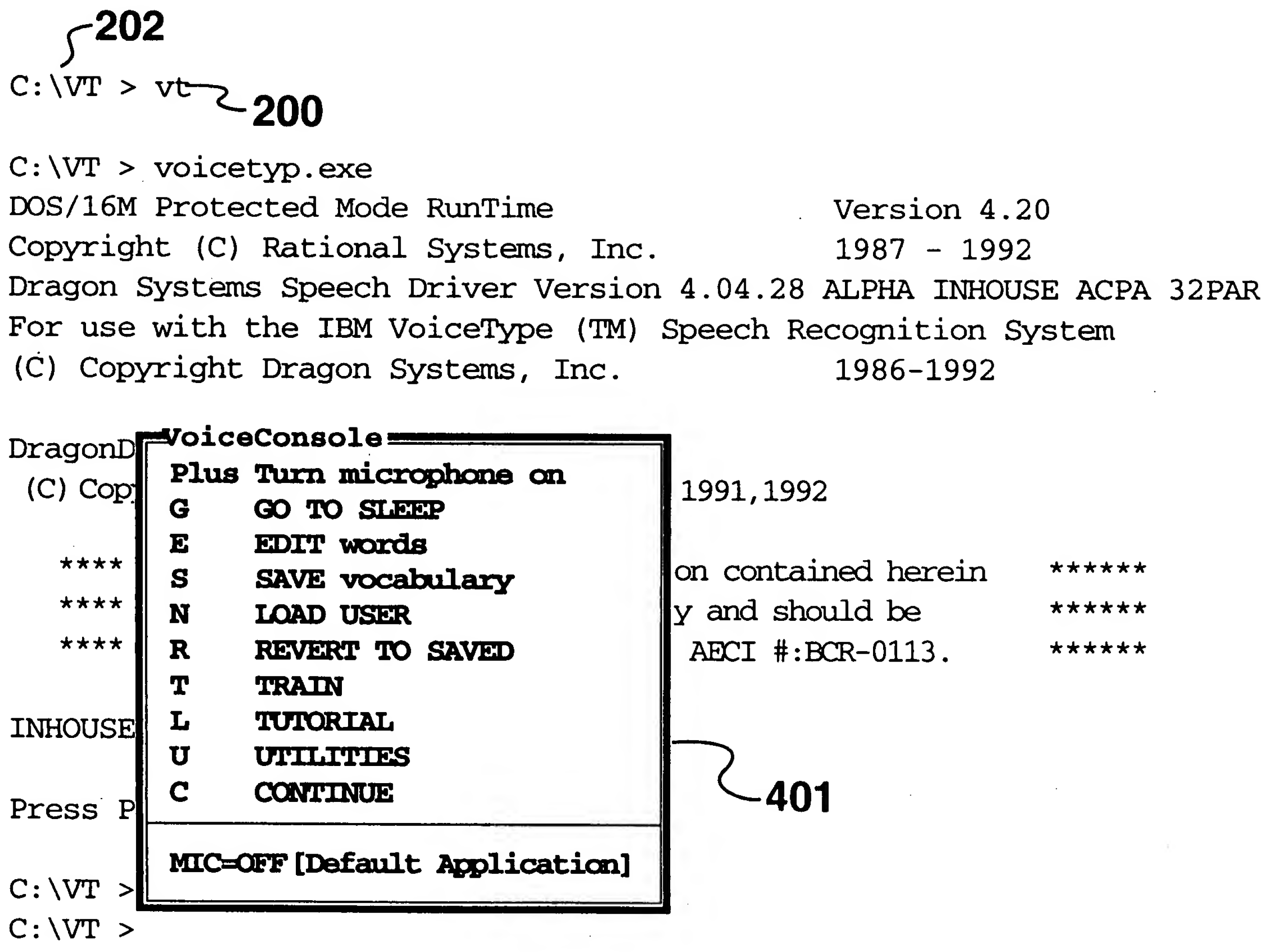
FIG. 20

```
-GET_ALLOWED_RESPONSE SUBROUTINE~180
    -SET UTTERANCE_NUMBER TO 0~526
    -UTTERANCE_LOOP: REPEAT UNTIL EXIT FROM WITHIN~528
        -INCREMENT UTTERANCE_NUMBER~530
        -WAIT FOR USER INPUT~532
        -IF KEYSTROKE, RETURN WITH KEY AS RESPONSE~534
        -CALL LARGE VOCABULARY RECOGNIZER TO SCORE UTTERANCE'S TOKEN
        AGAINST LARGE VOCABULARY, REQUESTING SCORE OF BEST SCORING 25
        WORDS~536
        -SET USER_RESPONSE TO ZERO~538
        -WORD_LIST_LOOP: FOR EACH WORD RETURNED BY THE RECOGNIZER, IN
        ORDER OF SCORE WITH BEST SCORING FIRST~540
            -IF ITS SCORE IS WORSE THAN A GIVEN LEVEL~542
                -EXIT WORD_LIST_LOOP
            -IF IT IS AN ALLOWED RESPONSE WORD~546
                -SET USER_RESPONSE TO THE BEST SCORING ALLOWED
                RESPONSE WORD~548
```

FIG. 21

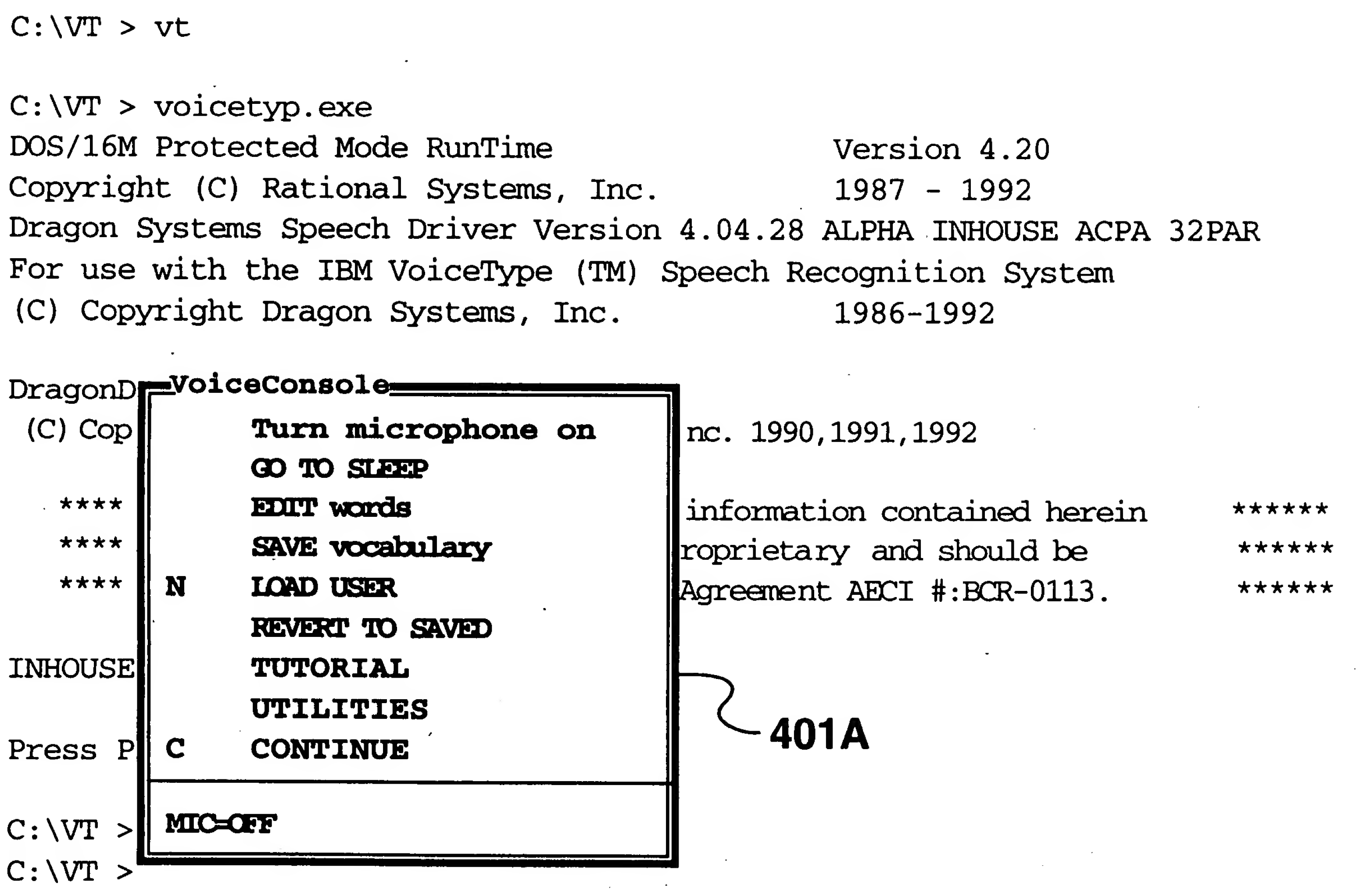
```
                              -CALL ADAPTIVE_TRAINING SUBROUTINE FOR
                              TOKEN, AND ANY SIMILAR TOKEN[X]s FROM
                              PREVIOUS LOOP, AND BEST SCORING
                              ALLOWED RESPONSE WORD, IF THAT WORD IS
                              THE EXPECTED WORD~550
                 -LABEL TOKEN WITH BEST SCORING ALLOWED RESPONSE WORD,
                 IF THAT WORD IS THE EXPECTED WORD~552
                 -RETURN~553
       -IF USER_RESPONSE IS ZERO~554
            -SAVE TOKEN AS TOKEN[UTTERANCE_NUMBER]~556
            -IF UTTERANCE_NUMBER = 1~558
                 -PROMPT USER TO REPEAT WHAT JUST SAID
            -OTHERWISE~560
                 -PROMPT USER TO SAY EXPECTED WORD~562
                 -IF UTTERANCE_NUMBER >2~564
                      -COMPARE TOKEN[X]s WITH EACH OTHER~566
                      -IF THREE SCORE WITHIN A GIVEN DISTANCE OF EACH
                      OTHER~568
                           -LABEL THE THREE CLOSELY SCORING TOKEN[X]s
                           WITH EXPECTED WORD~570
                           -SET USER_RESPONSE TO EXPECTED WORD~572
                           -EXIT UTTERANCE_LOOP~574
                      -ELSE IF UTTERANCE_NUMBER = 5,~576
                           -LABEL THREE TOKEN[X]s WHICH COMPARE MOST
                           CLOSELY AS EXPECTED WORD~578
                           -SET USER_RESPONSE TO EXPECTED WORD~580
                           -EXIT UTTERANCE_LOOP~582
  -IF USER_RESPONSE IS NOT ZERO~584
       -CALL ADAPTIVE TRAINING SUBROUTINE FOR UTTERANCE'S THREE BEST
       SCORING TOKEN[X]s AND EXPECTED WORD~
       -SAVE THREE CLOSEST TOKEN[X]s, LABELED BY THEIR ASSOCIATED
       EXPECTED WORD~585
```

FIG. 21 CONT.



```
C:\VT > vt

C:\VT > voicetyp.exe
DOS/16M Protected Mode RunTime                    Version 4.20
Copyright (C) Rational Systems, Inc.              1987 - 1992
Dragon Systems Speech Driver Version 4.04.28 ALPHA INHOUSE ACPA 32PAR
For use with the IBM VoiceType (TM) Speech Recognition System
(C) Copyright Dragon Systems, Inc.                1986-1992

DragonD ═VoiceConsole══════════════
 (C) Cop ║Plus Turn microphone on   ║ 1991,1992
         ║G    GO TO SLEEP          ║
   ****  ║E    EDIT words           ║on contained herein    ******
   ****  ║S    SAVE vocabulary      ║y and should be        ******
   ****  ║N    LOAD USER            ║ AECI #:BCR-0113.      ******
         ║R    REVERT TO SAVED      ║
         ║T    TRAIN                ║
INHOUSE  ║L    TUTORIAL             ║
         ║U    UTILITIES            ║
Press P  ║C    CONTINUE             ║
         ║──────────────────────────║
C:\VT >  ║MIC=OFF [Default Application]
C:\VT >  ╚══════════════════════════╝
```

Figure 22



```
C:\VT > vt

C:\VT > voicetyp.exe
DOS/16M Protected Mode RunTime                 Version 4.20
Copyright (C) Rational Systems, Inc.           1987 - 1992
Dragon Systems Speech Driver Version 4.04.28 ALPHA INHOUSE ACPA 32PAR
For use with the IBM VoiceType (TM) Speech Recognition System
(C) Copyright Dragon Systems, Inc.             1986-1992

DragonD
 (C) Cop                                 nc. 1990,1991,1992

   ****                                  information contained herein      ******
   ****                                  roprietary and should be          ******
   ****                                  Agreement AECI #:BCR-0113.        ******

INHOUSE

Press P

C:\VT >
C:\VT >
```

```
VoiceConsole
        Turn microphone on
        GO TO SLEEP
        EDIT words
        SAVE vocabulary
N       LOAD USER
        REVERT TO SAVED
        TUTORIAL
        UTILITIES
C       CONTINUE

MIC=OFF
```

Figure 23

```
                                              471
     GetValidEvent( mask=1 )
        Globals:
472        TIMEOUT 40 (moff) (noclr) --> CALL global-mic-off
           ANYKEY (moff) (noclr) --> CALL global-mic-off
           ANYKEY (norm) (nxpg) (moff) (noclr) --> CALL global-unknown-key
           KEY 'Enter' (norm) (nxpg) (moff) (noclr) --> CALL global-key-not-now
           KEY 'KeyPadEnter' (norm) (nxpg) (moff) (noclr) --> CALL global-key-not-now
           ANYSPELLKEY (norm) (nxpg) (moff) (noclr) --> CALL global-key-not-now
           KEY '+' (norm) (nxpg) (moff) (noclr) --> CALL global-wrong-plus-key
           KEY '-' (norm) (nxpg) (moff) (noclr) --> CALL global-wrong-minus-key
           TIMEOUT 40 (norm) (nxpg) (noclr) --> CALL global-timeout
           KEY 'F1' (norm) (nxpg) (moff) (noclr) --> CALL global-get-help
           UTT "[get help]" (norm) (nxpg) (noclr) --> CALL global-get-help
           UTT_TOO_LOUD (norm) (nxpg) (moff) (noclr) --> CALL global-too-loud
           UTT_TOO_SOFT (norm) (nxpg) (moff) (noclr) --> CALL global-too-soft
           REJECTED_UTT (norm) (nxpg) (moff) (noclr) --> CALL global-rejected-utt
           UTT_STRANGE (norm) (nxpg) (moff) (noclr) --> CALL global-rejected-utt
           TALK_TOO_FAST (norm) (nxpg) (moff) (noclr) --> CALL global-talk-too-fast
           UTT_TOO_LONG (norm) (nxpg) (moff) (noclr) --> CALL global-utt-too-long
           KEY 'Esc' (norm) (nxpg) (moff) (noclr) --> CALL global-escape
           KEY 'Minus' (norm) (nxpg) (moff) (svmsg) --> CALL global-mainmenu
           UTT "[Tutor menu]" (norm) (nxpg) (svmsg) --> CALL global-mainmenu
           KEY 'Plus' (norm) (nxpg) (moff) (svmsg) --> CALL global-voice-console
           UTT "[voice console]" (norm) (nxpg) (svmsg) --> CALL global-voice-console
474     Defaults:
           LASTWORD "[new paragraph]" (norm) (noclr) --> CALL default-lastword
           NEXTWORD ", "comma"" (norm) (noclr) --> CALL default-nextword
           LASTWORD "[new paragraph]" (nxpg) (noclr) --> CALL default-nextpage
           CURWORD "down" (nxpg) (noclr) --> CALL default-nextpage
           NEXTWORD ", "comma"" (nxpg) (noclr) --> CALL default-nextpage
           KEY 'F2' (norm) (noclr) --> CALL default-no-function-keys
           KEY 'F3' (norm) (noclr) --> CALL default-no-function-keys
           KEY 'F4' (norm) (noclr) --> CALL default-no-function-keys
           KEY 'FS' (norm) (noclr) --> CALL default-no-function-keys
           KEY 'F6' (norm) (noclr) --> CALL default-no-function-keys
           KEY 'F7' (norm) (noclr) --> CALL default-no-function-keys
           KEY 'F8' (norm) (noclr) --> CALL default-no-function-keys
           KEY 'F9' (norm) (noclr) --> CALL default-no-function-keys
           KEY 'F10' (norm) (noclr) --> CALL default-no-function-keys
           UTT "[oops!]" (norm) --> CALL d3gd-oops
476     Cases:
           UTT "down" (norm) --> *e
           UTT "[choose 1 ]" (norm) --> GOTO d2gd-said-okay
           UTT "[okay]" (norm) --> GOTO d2gd-said-okay
           KEY 'Backspace' (norm) --> CALL d2gd-ignore-backspace
           LASTSPELLKEY '[' (norm) --> CALL d2gd-one-right
477        ANYSPELLKEY (norm) --> CALL d2gd-one-wrong
        Ceiling:
     End of Stack.
```

FIG. 28

```
     ******************************************************************
     ** MODULE NAME: final7.pln
     ** Copyright (c) Dragon Systems,Inc. 1992
     ** OWNER:        Joel Gould
     ** CREATED:      September 4, 1992
     ** FUNCTIONS
     ** DESCRIPTION
502  ** Chapter 7
     ** This topic teaches the user to correct dictation errors by
     ** selecting words from the choice list.
     ****************************************************************
     ** MODIFICATIONS
     ** ...
     **
     ****************************************************************
504  CHAPTER Correcting Dictation with the Choice List

     DEFAULT NOCLEAR LASTWORD           CALL default-lastword
     DEFAULT NOCLEAR NEXTWORD           CALL default-nextword
     DEFAULT NEXTPAGE NOCLEAR LASTWORD  CALL default-nextpage
     DEFAULT NEXTPAGE NOCLEAR CURWORD   CALL default-nextpage
     DEFAULT NEXTPAGE NOCLEAR NEXTWORD  CALL default-nextpage
     DEFAULT NOCLEAR 'F2'                    CALL default-no-function-keys
     DEFAULT NOCLEAR 'F3'                    CALL default-no-function-keys
     DEFAULT NOCLEAR 'F4'                    CALL default-no-function-keys
506  DEFAULT NOCLEAR 'F5'                    CALL default-no-function-keys
     DEFAULT NOCLEAR 'F6'                    CALL default-no-function-keys
     DEFAULT NOCLEAR 'F7'                    CALL default-no-function-keys
     DEFAULT NOCLEAR 'F8'                    CALL default-no-function-keys
     DEFAULT NOCLEAR 'F9'                    CALL default-no-function-keys
     DEFAULT NOCLEAR 'F10'                   CALL default-no-function-keys
     DEFAULT NOCLEAR ANYSPELLKEY        CALL default-no-spelling-keys

     * IF INORDER GOTO chap7-start

508  EDITOR RESET

     * LESSON chap7-start

510  CONSOLE MIC ON
     CONSOLE SLEEP OFF
512  PROMPT HIDE
514  EDITOR SHOW
     *--------------------------------------------------------------
     `{HIGH}TOPIC: CORRECTING DICTATION WITH THE CHOICE LIST{NORM}
     `
     `This topic describes how to use the choice list to correct dictation
516  `errors. You are going to learn how to:
     `
     `  \b Accept {NAMENORM}'s default choice
     `
     `  \b Choose another word from the choice list
```

FIG. 30

```
    `Please say {SAY}"[okay]" to continue.{CR}
    `Please say {UTT}"[Tutor menu]" to display the menu.

518 EXPECTING "[okay]"

    IF INORDER CALL chap7-bonus-text

590 PROMPT RESET
    PROMPT SHOW
    PROMPT HIGHLIGHT OFF
    *
    *

    * PROMPT /when/suddenly/a/white/rabbit/with/pink/eyes/
    * PROMPT /ran/close/by/her/. \"period\"/
592 PROMPT /[new paragraph]/
    PROMPT /there/was/nothing/so/very/remarkable/in/that/; \"semicolon\"/
    PROMPT /nor/did/Alice/think/it/so/very/much/out/of/the/way/to/
    PROMPT /hear/the/rabbit/say/to/itself/, \"comma\"/" \"open quote\"/
    PROMPT /oh/dear/! \"exclamation point\"/oh/dear/! \"exclamation point\"/
    PROMPT /I/shall/be/too/late/! \"exclamation point\"/" \"close quote\"/
    PROMPT /( \"open paren\"/when/she/thought/it/over/afterwards/, \"comma\"/
    PROMPT /it/occurred/to/her/that/she/ought/to/have/wondered/at/this/, \"co
    PROMPT /but/at/the/time/it/all/seemed/quite/natural/) \"close paren\"/;
    \"semicolon\"/

596 PROMPT HIGHLIGHT ON

598 `Since you are starting a new topic, please start a new paragraph
    `in your document. Say {SAY}"[new paragraph]".

602 EXPECTING "[new paragraph]"
604 CHOICELIST 1="[new paragraph]"

606 `Please begin dictating this lesson by saying the first word in the
    `Text Prompter, {SAY}"there".

610 EXPECTING "there"
612 CHOICELIST 1="there"

616 `This is a choice list, which has appeared every time you've dictated a w
    `
    `If the word you said is correctly identified, it is listed first on the
    `choice list. However, you still have to tell {NAMENORM} that this
    `recognition is correct.
    `
    `There are three ways to do this.

620 NEWPAGE

622 `The first is to say the next word. This
```

FIG.30 CONT.-1

```
622  `is the method you used in the previous topic.
     `
     `The second way is to say {UTT}"[okay]". You used this method in earlier
     `topics.
     `
     `The third way is to say {UTT}"[choose 1]", since you want to choose
     `the first word on the choice list.

626  NEWPAGE

630  `Until now, the word the Text Prompter asked you to dictate has always
     `appeared as the first word on the choice list. But that doesn't always h
     `when you dictate in {NAMENORM}.
     `
     `Sometimes the word you dictate will be an alternate choice on the list.
     `
     `Sometimes it won't be on the list at all.
     `
     `Please continue dictating from the Text Prompter, starting with
     `{SAY}"was".

636  EXPECTING "was"
     *---------------------------------------------------------------------
638  call dictate1-no-error   * next: "nothing"
640  call dictate1-no-error   * next: "so"

652  `Sometimes {NAMENORM} identifies the word said as a possibility, but
     `not as the most likely choice. When this happens, the word will appear
     `on the choice list, but not as the first choice.
     `
     `Please dictate the next word.

656  call dictate1-no-error   * next: "very"
     *---------------------------------------------------------------------
660  CHOICELIST 1="vary" 3="very"
666  POINTAT CHOICELIST 3

668  `Although you said {UTT}"very", {NAMENORM} thought
     `that the most likely thing that you said was {UTT}"vary".
     `
     `{NAMENORM} learns from its mistakes and adapts to your style of speech.
     `Therefore, you must correct any recognition errors immediately.

     NEWPAGE

672  `If you fail to correct {NAMENORM}'s mistake in this case, every time you
     `{UTT}"very", it will type {UTT}"vary". If this mistake goes by undetecte
     `other words are also affected.
     `The next time you say {UTT}"merry", {NAMENORM} may think you mean
     `{UTT}"marry".

     NEWPAGE
```

FIG.30 CONT.-2

```
676 `If, as in this case, the word you spoke is not in the first position
    `on the choice list, you must tell {NAMENORM} which word you actually
    `spoke. You do this with the {UTT}"[choose n]" command, where {UTT}"n" re
    `the number of the word on the choice list.

    NEWPAGE

680 `In this case, you want {NAMENORM} to select the third word.
    `
    `Please say {SAY}"[choose 3]" now.

684 CASE {NEXTWORD} CALL must-say-choose-n
686 EXPECTING "[choose 3]"
    *-----------------------------------------------------------------------
688 CHOOSE 3

692 `Saying {UTT}"[choose 3]" made {NAMENORM} erase the word {UTT}"vary"
    `from the text and type the word {UTT}"very" instead.
    `
    `Because you chose the word you spoke, {NAMENORM} no
    `longer needs to show a list of possible interpretations of the utterance
    `and it has removed the choice list from the screen.
    `
    `As soon as you say the next word, the choice list will re-appear with a
    `new set of possibilities.

    NEWPAGE

696 `For the rest of this tutorial, the {NAMENORM} Tutorial will allow
    `random recognition errors
    `to occur while you practice your dictation.  Correct them as soon as
    `they happen, to prevent corruption
    `of your vocabulary.
    `
    `If {NAMENORM} correctly identifies the word you say, continue on to
    `the next word.  If it incorrectly identifies the word you say, correct i
    `by saying {UTT}"[choose n]", where {UTT}"n" is the number of the desired
    `word on the choice list. If you don't correct your errors, the Tutorial
    `will remind you.
    `
    `To start dictating again, please say the next word on your
    `Text Prompter, {SAY}"remarkable".

700 EXPECTING "remarkable"
    *-----------------------------------------------------------------------
702 call dictate1-no-error              * next: "in"
708 call dictate1-no-error              * next: "that"

714 CHOICELIST 1={CURWORD}

720 Please say {SAY}"; \"semicolon\"".
```

FIG.30 CONT-3

```
724 CASE "[choose 1]" CALL dlgd-said-okay
    CASE "[okay]" CALL dlgd-said-okay
726 EXPECTING "; \"semicolon\""

728 call dictate1-no-error              * next: "nor"
    *----------------------------------------------------------------
    *must-correct-errors

    `Notice that the word "nor" did not appear first on
734 `your choice list. Please choose the correct word now,
    `and then continue dictating.

    *----------------------------------------------------------------
738 CALL dictate1-on-list               * next: "did"

    CHOICELIST 1={CURWORD}
762 CASE "[choose 1]" CALL dlgd-said-okay
    CASE "[okay]" CALL dlgd-said-okay
    EXPECTING "Alice"
    *----------------------------------------------------------------
    *                                    expecting:
    *                                   ------------
766 CALL dictate1-no-error              * think
768 CALL dictate1-no-error              * it
770 CALL dictate1-on-list               * so
    CALL dictate1-no-error              * very
    CALL dictate1-no-error              * much
    CALL dictate1-on-list               * out
    CALL dictate1-on-list               * of
    CALL dictate1-no-error              * the
    CALL dictate1-no-error              * way
    CALL dictate1-no-error              * to
    CALL dictate1-no-error              * hear
    CALL dictate1-no-error              * the
    CALL dictate1-on-list               * rabbit
    CALL dictate1-no-error              * say
    CALL dictate1-no-error              * to
    CALL dictate1-on-list               * itself
    CALL dictate1-no-error              * , \"comma\"
    CALL twoword1-open-quote            * " \"open quote\"
                                        * oh
    CALL dictate1-no-error              * dear

    `{NAMENORM} has two words for the {UTT}'!' character:
    `{UTT}"! \"exclamation point\"" and
    `{UTT}"! \"exclamation mark\"".
    `
    `While you use the {NAMENORM} Tutorial, however,
    `only {UTT}"! \"exclamation point\"" is active.
    `
```

FIG.30 CONT-4

```
************************************************************************
** MODULE NAME: dictate.pln
** Copyright (c) Dragon Systems,Inc. 1992
** AUTHOR:      Joel Gould
** CREATED:     Sept 17, 1992
** FUNCTIONS
** DESCRIPTION
** {NAMESHORT} Trainer lesson plan component
** -Originally part of global.pln, this file contains the lesson plan
** code which handles dictation practice
** ...
*************************************************************************
* MODIFICATIONS
** ...
************************************************************************
*
* DICTATION PRACTICE SUBROUTINE - 1
*
* Includes support for
*    - choose words
*
* Each subroutine should be called for one word in the teleprompter.
* Just before calling the subroutine should be an EXPECTING command
* for the word in question.  Each subroutine will end with an EXPECTING
* command and return only if the next word in the teleprompter was
* spoken.
*
* For example:
*
* PROMPT /one/two/three/four/
* EXPECTING "one"
* CALL dictate1-no-error    * one is 1st on choice list; expecting two
* CALL dictate1-no-error    * two is 1st on choice list; expecting three
* CALL dictate1-on-list     * three is put in random slot on choice list
*                 *           upon exit we will be expecting four
* CHOICELIST 1="four"
*
************************************************************************
*
* ---> DICTATE1-RANDOM
*
* Currently forces an on-list error if we just had a misrecognition.
* Also introduces errors 5% of the time (just to be sure we get one)
*
LESSON dictate1-random
IF SHORTWORD GOTO dictate1-no-error
RANDOMIZE 50 dictate1-no-error
IF MISRECOG  GOTO dictate1-on-list
RANDOMIZE  5 dictate1-on-list
GOTO dictate1-no-error
*-----------------------------------------------------------------------
************************************************************************
```

FIG.31

08/382,752

```
      *
      * ---> DICTATE1-NO-ERROR
      *
      * Put current word first on choice list, then get the next word
      *
640~  LESSON dictate1-no-error
640A~ CHOICELIST 1={CURWORD}
640B~ LESSON dictate1-no-error-after
640C~ HIGHLIGHT NEXTWORD                   * LASTWORD <- CURWORD
640D~ CASE "[okay]"       GOTO d1gd-said-okay
640E~ CASE "[choose 1]"   GOTO d1gd-said-okay
640F~ EXPECTING {CURWORD}
640G~ RETURN
      *-----------------------------------------------------------------------
      * We end up here if the user has said OKAY or something else which
      * accepts the last word and clears the choice list.  Here we expect
      * him, to say the next word.
      *
646~  LESSON d1gd-said-okay
646A~ CHOOSE {LASTWORD}
646B~ EXPECTING {CURWORD}
646C~ RETURN
      *-----------------------------------------------------------------------


      ************************************************************************
      *
      * ---> DICTATE1-ON-LIST
      *
      * Pick a random slot for the word to appear which is not the first
      * slot on the choice list.  Make sure the user says "choose-N",
      * then get the next word
      *
740~  LESSON dictate1-on-list
740A~ CHOICELIST ?={CURWORD}
740B~ HIGHLIGHT NEXTWORD                   * LASTWORD <- CURWORD
740C~ CASE {CURWORD}    CALL d1on-say-choose-n
740D~ CASE "[okay]"     CALL d1on-say-choose-n
740E~ CASE "[choose 1]" CALL d1on-say-choose-n
740F~ EXPECTING "[choose {?}]"
740G~ CHOOSE {?}
740H~ EXPECTING {CURWORD}
740I~ RETURN
      *-----------------------------------------------------------------------
746~  LESSON d1on-say-choose-n
746A~ AFTERSEEN 1 d1on-short1-say-choose-n

      `The performance of {NAMESHORT} improves with every error it makes,
746B  `but only if you correct the mis-recognitions.  If you do not correct
      `every error, {NAMESHORT}'s performance will get worse.
      `
      `{NAMESHORT} has incorrectly identified the word you just spoke.
```

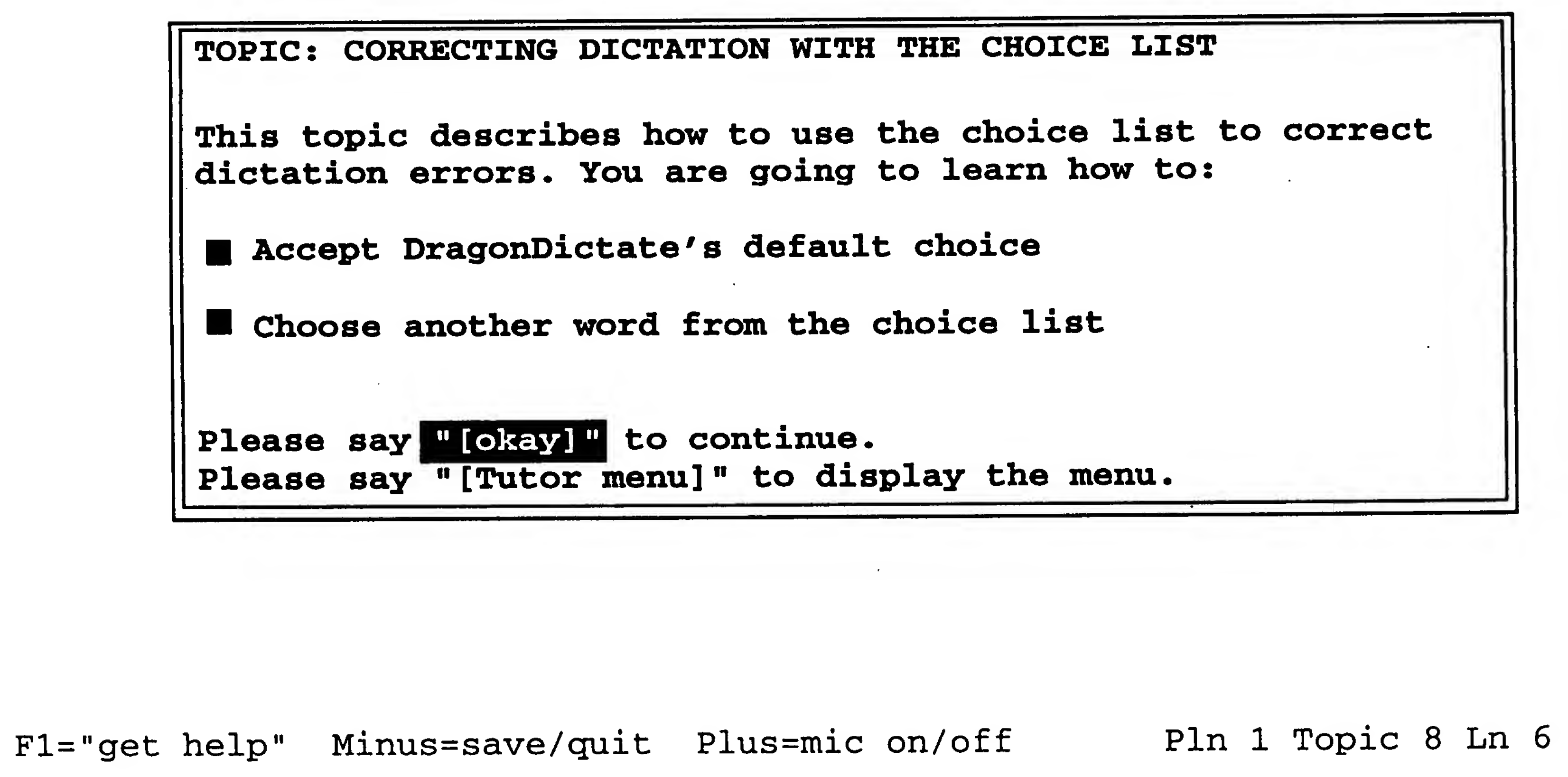
FIG.31 CONT.-1

```
       `The correct word {UTT}{LASTWORD} is on the choice list, however,
       `and you can correct {NAMESHORT}'s mis-recognition. Please {WHAT2DO}.

746D ~REMOVEUTT
746E ~RETURN REPEAT
      *-----------------------------------------------------------------------
748  ~LESSON dlon-short1-say-choose-n
748A ~RANDOMIZE 25 dlon-short2-say-choose-n
748B ~RANDOMIZE 33 dlon-short3-say-choose-n
748C ~RANDOMIZE 50 dlon-short4-say-choose-n

748D { `Please correct {NAMESHORT}'s mis-recognition before continuing.
       `Please {WHAT2DO}.

748E ~REMOVEUTT
748F ~RETURN REPEAT
      *-----------------------------------------------------------------------
750  ~LESSON dlon-short2-say-choose-n

      `Please {WHAT2DO} to correct that last mis-recognition.

      REMOVEUTT
      RETURN REPEAT
      *-----------------------------------------------------------------------
752  ~LESSON dlon-short3-say-choose-n

      `It is very important to correct all mis-recognitions to
      `prevent your vocabulary files from being corrupted.
      `Please say {SAY}{EXPECTED}.

      REMOVEUTT
      RETURN REPEAT
      *-----------------------------------------------------------------------
754  ~LESSON dlon-short4-say-choose-n

      `Correct the last error before continuing to dictate.

      REMOVEUTT
      RETURN REPEAT
      *-----------------------------------------------------------------------
```

FIG.31 CONT-2

```
      * ---> DICTATE3-RANDOM
      *
      * Currently forces an on-list error if we just had a misrecognition.
      * Also introduces errors 5% of the time (just to be sure we get one)
      *
      * When an error is indicated, we choose on-list 60% of the time and
      * off-list 40% of the time.
      *
779~  LESSON dictate3-random
779A~ IF SHORTWORD GOTO dictate3-no-error
779B~ RANDOMIZE 50 dictate3-no-error
779C~ IF MISRECOG  GOTO d3-error
779D~ RANDOMIZE 5  d3-error
779E~ GOTO dictate3-no-error
      *------------------------------------------------------------------
779F~ LESSON d3-error
779G~ RANDOMIZE 60 dictate3-on-list
779H~ GOTO dictate3-off-list
      *------------------------------------------------------------------
```

FIG.31 CONT.-3



TOPIC: CORRECTING DICTATION WITH THE CHOICE LIST

This topic describes how to use the choice list to correct dictation errors. You are going to learn how to:

- Accept DragonDictate's default choice
- Choose another word from the choice list

Please say "[okay]" to continue.
Please say "[Tutor menu]" to display the menu.

```
F1="get help"  Minus=save/quit  Plus=mic on/off       Pln 1 Topic 8 Ln 6
```

Figure 32

```
-Initialization()~1002
     -...
     -take start time~1008
     -run integer tasks~1010
     -take end time~1012
     -subtract start time from end time to get task duration~1014
     -set NumberToPassPrefilter and ScoreThreshold in correspondence to
     task duration~1016
     -...
     -detect if DSP board is present~1018
     -if DSP board is not present, set DSPBoardPresent to false~1020
     -else~1021
          -set DSPBoardPresent to true~1022
          -download DSP code to DSP board~1024
          -initialize DSP board~1026
     -...
     -call MSW SetWindowsHookEx with WH_CALLWNDPROC to set hook for
     CallWndProc procedure that monitors menu messages~1028
     -call MSW SetWindowsHookEx with WH_KEYBOARD to set hook for
     KeyboardProc procedure that monitors keystrokes~1030
     -initialize and clear MenuStack~1034
     -initialize and clear HWndToAppTable~1038
     -display the VoiceBar~1042
     -set RecognizerOn to true~1044
     -set ChoiceListOperative to false~1046
     -...
```

FIG. 47

```
-DSP board code~1025
     -...
     -every 1/100 second~1050
          -perform utterance detection~1052
          -if detect utterance, notify host~1054
          -...
          -increment OddEvenCount~1056
          -calculate an FFT of the last 1/100 second of audio signal~1058
          -calculate the Cepstrum of the last 1/100 second of audio
          signal~1060
          -place the FFT and selected Mel Cepstrum values into a frame
          format~1062
          -if OddEvenCount is even save the just calculated frame~1064
          -if OddEvenCount is odd~1066
               -add the individual values of the just calculated frame to
               the corresponding values of the frame saved in the
               previous 1/100 second~1068
               -divide each value in the frame by two~1070
               -send the averaged frame, representing FFT and Mel
               Cepstrum values for last 1/50 second, to the host
               processor for addition to the frame buffer~1072
     -...
```

FIG. 48

```
-CallWndProc(code, wParam, lParam)~1029
    -if message is WM_INITMENU, indicating a menu is about to become
    active~1664
        -clear MenuStack~1666
        -place a MenuEntry with the MenuHandle indicated by WM_INITMENU
        in the MenuStack~1668
    -if message is WM_INITMENUPOPUP, indicating a popup menu is about to
    become active~1670
        -if a MenuEntry with the pop-up menu's menu handle in the
        MenuHandle field is not currently at the end of the MenuStack,
        add such an entry and place in the preceding entry in the
        MenuStack the MenuItemID corresponding to the item in the
        parent menu from which the popup menu came~1672
    -if message is WM_MENUSELECT, indicating a user has selected a menu
    item~1674
            -scan the MenuStack for an entry with MenuHandle matching
            that in the WM_MENUSELECT message~1676
            -if find a match~1678
                -if find the match other than at the end of the
                MenuStack, delete the MenuEntries after the matching
                MenuEntry from the stack~1680
                -record the menu item ID returned by WM_MENUSELECT in
                the MenuItemID field of the MenuEntry with the
                matching MenuHandle~1682
            -else use calls to MSW GetSubMenu to do a tree search,
            starting with menu handle returned by GetMenu, until find
            the menu with selected item, and then reestablish the
            MenuStack with the path in the menu tree which leads to
            menu of the selected item.~1684
    -if message is WM_NCDESTROY, indicating a window is being
    closed~1686
        -if WM_NCDESTROY is being sent to a window having a handle in
        the HWndToAppTable, delete that handle's entry in table~1688
    -if message is WM_ACTIVATE, indicating a window is being
    activated~1690
        -call ApplicationTracking with the window's HWnd~1692
        -pop-up any key alteration windows, if any, appropriate for the
        new active window~1693
    -if message is WM_CREATE, indicating a window is being created~1694
        -if the new window's handle is already in HWndToAppTable,
        delete the handle's entry in table~1696
    -if message is WM_SHOWWINDOW, indicating a window that was
    previously covered is being uncovered~1698
        -if a call to MSW GetWindow with GW_OWNER for the window
        indicates it is a application window or a dialog window, call
        ApplicationTracking with the window's HWnd~1700
    -return~1702
```

FIG. 49

```
-KeyboardProc(code, wParam, lParam)~1032
     -...
     -if ChoiceListOperative is true and the last message group header
     before the read pointer in the JournalPlaybackProc's message queue
     indicates the current message group was created for a word
     recognized from the "Choice List" state~1033
          -use MSW PostMessage to send keystroke information represented
          by wParam and lParam to ChoiceList~1035
          -return with indication the keystroke message which caused
          KeyboardProc to be called should be discarded~1037
     -...
```

FIG. 50

```
-MenuStack~1036
     -list of MenuEntry structs~1854, each containing
          -MenuHandle~1856
          -MenuItemID~1858
```

FIG. 51

```
-HWndToAppTable~1040
     -a list of entry structs each containing~1654
          -HWnd~1656
          -AppState~1658
          -AppMode~1660
          -ShiftKeyOn~1704
          -ControlKeyOn~1706
          -AltKeyOn~1708
```

FIG. 52

```
-FastDemon()~1048
     -if DSPBoardPresent is true~1074
          -if RecognizerOn is false~1076
               -if the DSP board is on, stop it~1078
          -else~1080
               -if the DSP board is stopped, start it~1082
               -if have received notification of an utterance detection
               from the DSP board, call RecSetupCallAndOutput for the
               utterance~1083
     -else if DSPBoardPresent is false~1084
          -if RecognizerOn is true~1086
               -perform incremental utterance detection on new signals in
               audio buffer~1088
               -if an utterance is detected, call RecSetupCallAndOutput
               for the utterance~1090
               -while there is more than 1/50 of a second of audio signal
               in the audio buffer~1092
                    -for every 1/50 second of the signal~1094
                         -calculate its FFT and Cepstrum~1096
                         -place the FFT and selected Mel Cepstrum values
                         into a frame format~1098
                         -add the frame to end of a frame buffer~1100
     -if  choice list is displayed and ChoiceListOperative is false~1104
          -increment DelayCount~1106
          -if DelayCount is => ChoiceListRemovalDelay, remove display of
          choice list~1108
     -...
```

FIG. 53

```
-RecSetupCallAndOutput(Utterance)~1102
     -if CurrentMode is BaseVocabSelectMode~1154
          -clear StateList and then place in it the state having versions
          of the PromptedWord from each base vocabulary~1156
          -call Recognize for the utterance with current StateList and
          with LanguageContext and StartString Nulled~1158
          -use MSW PostMessage to send BaseVocabSelection routine a
          PromptedUtterance message, with a pointer to the recognition
          results, including the recognition's score for each of the
          words from the PromptedWord's corresponding state~1160
          -return~1162
     -else if CurrentMode is TrainWordMode~1164
          -clear StateList and then place PromptedWord in it~1166
          -if the PromptedWord is not a word listed in the "Train Word"
          state and if OnlyListenForWordsBeingTrained is false, add the
          "Train Word" state to the StateList~1168
          -call Recognize for the utterance with the current StateList,
          and with LanguageContext and StartString NULLed~1170
          -use MSW PostMessage to send TrainWordDialog a
          PromptedUtterance message, with a pointer to the recognition
          results and with a pointer to the recognition's utterance~1172
          -return~1174
     -else if CurrentMode is CommandMode or DictateMode~1176
          -clear StateList and then add the it the "Always Active" and
          "Global Commands" states~1178
          -if a call to MSW GetSystemDebugState returns SDS_MENU
          indicating a menu is currently active~1180
               -set CurrentMode to CommandMode~1182
          -else~1184
               -call ApplicationTracking with a Null HWnd to get the
               current entry in the HWndToAppTable~1186
               -set CurrentAppState and CurrentMode equal to the AppState
               and AppMode in the table entry returned~1188
               -add CurrentAppState to StateList~1190
          -if CurrentMode is DictateMode~1192
               -if ChoiceList routine has not been initialized,
               initialize it~1193
               -if ChoiceListOperative is true add "Choice List" state
               StateList~1194
               -add "DictateMode" state to StateList~1196
               -call LanguageContextTracking to set the current
               LanguageContext~1198
          -if CurrentMode is CommandMode~1200
               -call CommandTracking to set the CurrentTrackingState~1202
               -add the CurrentTrackingState to the StateList~1204
               -set LanguageContext to Null~1206
          -call Recognize for the utterance with its associated
          LanguageContext and StateList and with StartString Null~1208
```

FIG. 54

```
    -store the utterance just recognized, and the LanguageContext
    and StateList for the utterance, and its up to nine best
    scoring words and their associated states in a
    WordHistoryBuffer~1210
    -call PerformWordsOutput for the best scoring word, its
    associated state, and pointer into utterance's entry in
    WordHistoryBuffer, if any~1212
    -return~1214
-...
```

FIG. 54 CONT.

```
-Recognize(Utterance, LanguageContext, StateList, StartString)~1110
     -if StartString is not empty, limit active vocabulary to words in
     states of StateList which start with the letters of the StartString,
     independent of case~1114
     -if CurrentMode is DictateMode add an initial language context
     component, which depends in part from LanguageContext, to each
     prefilter score~1116
     -score the prefilter start of each word model in the entire
     vocabulary~1118
     -limit active word model candidates to the NumberToPassPrefilter
     words with best scoring prefilter scores, ensuring that all of the
     words in the active vocabulary up to the NumberToPassPrefilter are
     included~1120
     -for each active word model candidate~1122
          -if it is a helper model, create in RAM a list of pointers to
          the PELs listed in that model~1124
          -else if it is a phonetic model,~1126
               -create an empty PEL pointer list in RAM for the
               model~1128
               -for each phoneme in its phonetic spelling~1130
                    -define a corresponding PIC according to the phoneme
                    and its preceding phoneme or silence and its
                    following phoneme or silence~1132
                    -add to the model's PEL pointer list a pointer to
                    each PEL associated with that PIC~1134
     -for each successive frame of Utterance in frame buffer until
     scoring of all active word candidates is complete~1136
          -for each active word model candidate~1138
               -use the frame to update the relative score of the match
               of the word model against the frame sequence of the
               current Utterance~1140
               -if CurrentMode is DictateMode, if the match procedure
               makes a transition to one of the word models first four
               nodes, add a language context component, which depends in
               part from LanguageContext, to the score~1142
               -if the word model's score is worse than ScoreThreshold,
               remove it from the list of active word model
               candidates~1144
     -place word IDs of the up to NoOfWordsToReturn best scoring words
     from the active vocabulary which score above a given threshold, and
     their corresponding scores, in a results buffer~1146
     -for each such word ID, scan active states in the StateList in order
     of the state's priorities, to find the first state in which the Word
     ID occurs and place that state in association with the word's ID in
     the results buffer~1148
     -return with a pointer to the results buffer~1150
```

FIG. 55

1114A

```
-if StartString is not empty~2112
     -for each word in the states of the StateList~2114
          -add the word to the active vocabulary if its spelling
          contains a MatchingString which meets the following three
          conditions:~2116
               -each uppercase letter in StartString is matched by
               the same upper case letter in a corresponding
               position in the MatchString~2118
               -each lower case letter in in StartString is match by
               the same letter in either case in a corresponding
               position in the MatchString~2120
               -The MatchString starts the spelling of the word,
               except if the word's spelling contains a "[", the
               matching string can start immediately after the
               "["~2122
```

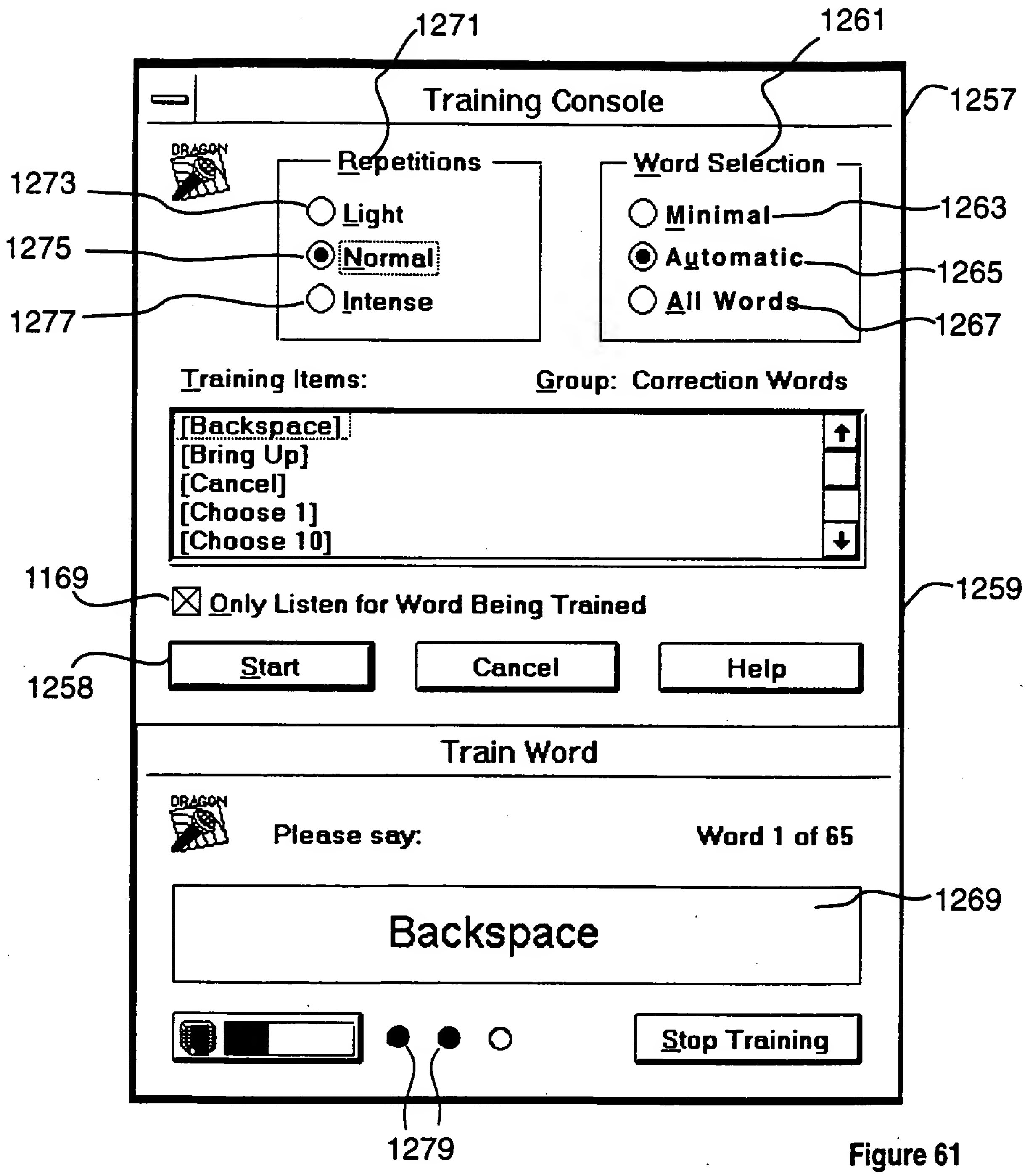
FIG. 55A

```
-BaseVocabSelection()~1216
     -display Create User dialog box and obtain up to eight character
     file name from the user~1218
     -display Identify Microphone dialog box and obtain description of
     user's microphone~1219
     -clear scores for each base vocabulary~1220
     -if user identifies a microphone type, weight scores of the base
     vocabularies associated with that microphone type~1222
     -load SELECTION.VOC and SELECTION.USR file~1224
     -display Sample Voice dialog box~1226
     -set CurrentMode to BaseVocabSelectMode~1228
     -for each word in prompted word list~1230
          -set PromptedWord equal to the current word~1232
          -prompt user to say PromptedWord by displaying it~1234
          -message loop~1236
               -call MSW GetMessage~1238
               -if receive PromptedUtterance message~1240
                    -add score associated with each base vocabulary's
                    version of the word to a total for that base
                    vocabulary~1242
                    -if the score of one of the base vocabularies exceeds
                    that of all the others by more than a specified
                    threshold, exit for loop~1244
                    -skip to for loops iteration for next word in
                    prompted word list~1246
     -select the base vocabularies whose associated word models have the
     best score~1248
     -...
     -create a new directory for the user~1250
     -create a copy of the selected base vocabulary's .USR file, with the
     pre-extension portion of its file name the name entered by the user,
     in the user's directory so the PIC table and PEL models in that .USR
     file will be used in the recognition of that user's utterances~1254
     -Set CurrentMode to CommandMode~1256
```

FIG. 56

```
-TrainWordDialog(WordList)~1256
     -display Train Word dialog box~1260
     -set CurrentMode to TrainWordMode~1262
     -for each active word on WordList~1264
          -set PromptedWord equal to the word's ID~1266
          -prompt user to say PromptedWord by displaying~1268
          -if the Repetitions button pressed is~1270
               -"Light": set MinRepetitions to 1 and MaxRepetitions to
               3~1272
               -"Normal": set MinRepetitions to 3 and MaxRepetitions to
               5~1274
               -"Intense": set MinRepetitions to 6 and MaxRepetitions to
               9~1276
          -display MinRepetitions unlit indicator lights~1278
          -set TokensForWord and GoodScoringTokensForWord both to
          zero~1280
          -message loop~1282
               -call MSW GetMessage~1284
               -...
               -if receive PromptedUtterance message~1286
                    -if the best scoring word in the recognition results
                    associated with the PromptedUtterance message is
                    other than the PromptedWord and if that best scoring
                    word has a score above a certain threshold, call
                    PerformWordsOutput for the best scoring word and its
                    associated recognized state~1287
                    -else, if the best scoring word in the recognition
                    result associated with PromptedUtterance message is
                    the PromptedWord and if it has a score above a
                    certain threshold~1288
                         -increment TokensForWord~1290
                         -save utterance associated with
                         PromptedUtterance message as a token for
                         PromptedWord~1292
                         -light first unlit indicator light~1294
                         -if score of utterance against the previous
                         model of PromptedWord is better than a specified
                         GoodScore threshold, increment
                         GoodScoringTokensForWord~1296
                         -if TokensForWord => MaxRepetitions or if
                         GoodScoringTokensForWord => MinRepetitions, exit
                         message loop~1298
                         -else if there is no unlit indicator light, add
                         one~1300
               -if [Alt+s}~1304
                    -remove Train Word dialog box~1306
                    -return~1308
               -...
          -call WordTraining Program subroutine for PromptedWord with
          utterances saved for that word~1310
     -remove Train Word dialog box~1312
     -return~1314
```

FIG. 60



Repetitions: Light [✓ Normal] Intense

Word Selection: Minimal [✓ Automatic] All Words

Training Items: Group: Correction Words

[Backspace]
[Bring Up]
[Cancel]
[Choose 1]
[Choose 10]

☒ Only Listen for Word Being Trained

Please say: Word 1 of 65

Backspace

Figure 61

```
-PerformWordsOutput(Word, State, WordHistoryBufferPointer)~1112
     -if ChoiceListOperative is true and the choice list is not the
     active window~1390
          -if State is not "Choice List"use MSW PostMessage to send
          RemoveChoiceList message to ChoiceList routine~1392
     -if Word has any ExtraData in its State~1394
          -if first byte in the ExtraData field indicates following bytes
          are DragonDictate script~1396
               -call MacroInterpreter to interpret the script~1398
               -return~1400
          -else if the first byte in the ExtraData field indicates the
          following bytes are to be fed to the JournalPlaybackProc~1402
               -copy the following ExtraData bytes to TextOutput~1404
     -else if Word has no ExtraData in its State~1406
          -copy the word's spelling (prior to " []", if any) to
          TextOutput~1408
     -if ShiftKeyOn is true for the currently active window~1410
          -capitalize first letter of TextOutput~1412
          -set ShiftKeyOn to false for the currently active window~1414
     -if ControlKeyOn is true for the currently active window~1416
          -replace first character of TextOutput with its control key
          equivalent~1418
          -set ControlKeyOn to false for the currently active window~1420
     -if AltKeyOn is true for the currently active window~1422
          -replace first character of TextOutput with its alt key
          equivalent~1424
          -set AltKeyOn to false for the currently active window~1426
     -copy a message group header, indicating whether or not the
     characters in TextOutput are associated with a word from the "Choice
     List" state, into the JournalPlaybackProc's message queue~1427
     -copy each character in TextOutput into the JournalPlaybackProc's
     message queue following the message group header~1428
     -call MSW SetWindowsHookEx with WH_JOURNALPLAYBACKPROC to install
     the hook for the JournalPlaybackProc~1430
     -if CurrentMode is DictateMode, and if the state of the best scoring
     word is other than "Choice List", use MSW PostMessage to send
     DisplayChoiceList message to ChoiceList routine with
     WordHistoryBufferPointer, which points to Word's associated the
     utterance just recognized in WordHistoryBuffer~1432
```

FIG. 64

```
-ChoiceList()~1393
     -...
     -message loop~1433
          -call MSW GetMessage~1435
          -if message is~1437
               -DisplayChoiceList message containing a pointer to a
               specified Utterance in WordHistoryBuffer~1439
                    -set ChoiceListOperative to true~1441
                    -if the choice list window is not displayed, display
                    it~1443
                    -display the up to nine best scoring words stored in
                    the utterance's entry in the WordHistoryBuffer in
                    numbered order~1445
                    -clear StartString~1447
               -a printable keystroke message~1449
                    -add the key, with its case, to StartString~1451
                    -call Recognize for ChoiceList's original utterance,
                    StateList, LanguageContext and current
                    StartString~1453
                    -if Recognize comes back with fewer than 9 words,
                    word search .VOC file and backup dictionary for words
                    which match StartString, independent of case, up to
                    the number of remaining unfilled slots in the
                    ChoiceList~1455
                    -if best scoring word does not match case of
                    StartString, designate StartString as first choice
                    word, and other words after it in choice order~1457
                    -re-display choice list with results of re-
                    recognition and word search, if any~1459
                    -use highlighting to indicate which letters of the
                    first choice word in ChoiceList belong to the
                    StartString~1461
               -a "Choose N" message~1463
                    -if there is an Nth word in ChoiceList~1465
                         -set ChoiceListOperative to false~1467
                         -remove display of ChoiceList~1469
                         -if first choice word stored in
                         WordHistoryBuffer for ChoiceList's current
                         utterance had a spelling output, output enough
                         keystrokes to delete keystrokes, if any,
                         associated with that prior spelling output~1471
                         -call PerformWordsOutput for Nth word and it
                         corresponding state if any~1475
                    -else beep for error~1477
               -RemoveChoiceList message~1479
                    -set ChoiceListOperative to false~1481
                    -set DelayCount to zero~1483
               -...
```

FIG. 65

```
-MacroInterpreter(MacroScript)~1382
     -create a MacroInstance for running of current MacroScript~1434
     -...
     -until reach end of the MacroScript~1386
          -find the next macro statement in the MacroScript~1438
          -if statement is~1440
               -"MenuPick[string]": call MenuPick subroutine for the
               string~1442
               -"ControlPick[string]": call ControlPick subroutine for
               the string~1444
               -"SpellMode": and if ChoiceListOperative is true~1446
                    -make choice list the active window~1448
                    -set CurrentMode to CommandMode~1450
               -"CommandMode":~1452
                    -set CurrentMode to CommandMode~1454
                    -set the AppMode associated with the currently active
                    window in HWndToAppTable to CommandMode~1456
               -"DictateMode":~1458
                    -set CurrentMode to DictateMode~1460
                    -set the AppMode associated with the currently active
                    window in HWndToAppTable to CommandMode~1462
               -"MicrophoneOff":~1464
                    -set RecognizerOn to false~1466
                    -set MicOffConfirmed to false~1468
               -"MicrophoneOn":~1470
                    -set RecognizerOn to true~1472
                    -set MicOffConfirmed to false~1473
               -"ShiftKey": set the ShiftKeyOn value in the currently
               active window's entry in the HWndToAppTable to true~1476
               -"ControlKey": set the ShiftKeyOn value in the currently
               active window's entry in the HWndToAppTable to true~1478
               -"AltKey": set the ShiftKeyOn value in the currently
               active window's entry in the HWndToAppTable to true~1480
               -...
     -delete current MacroInstance~1482
     -return~1484
```

FIG. 67

```
-JournalPlaybackProc(code, wParam, lParam)~1403
     -if code equals HC_GETNEXT~1487
          -copy the unread message element pointed to by, or following,
          the JournalPlaybackProc's read pointer to the location in
          memory pointed to by lParam~1488
     -else if code equals HC_SKIP~1489
          -increment the read pointer to the next unread message element,
          if there is one~1490
          -if the read pointer points past the last unread message
          element in the message queue~1492
               -call MSW UnhookWindowsHookEx for the JournalPlaybackProc
               to de-active its hook~1494
               -clear the message queue and zero the read and write
               pointers~1496
     -return~1498
```

FIG. 68

```
-WordTraining(Word, TokenList)~1311
     -if Word has one or more models~1502
          -if Word has more than one word model~1504
               -score each token in the TokenList against each of Word's
               word models~1506
               -associate each token with the word model against which it
               scores best~1508
          -else, associate each token with Word's single model~1510
          -for each of Word's pronunciations with which tokens have been
          associated ~1512
               -set GoodSpelledModelTokens and GoodHelperModelTokens to
               0~1516
               -if the pronunciation has a spelled model, call Training
               to adapt that spelled model with all the tokens associated
               with the pronunciation's phonetic or helper model, adding
               the number of such tokens that were successfully used to
               adapt the spelled model to GoodSpelledModelTokens~1518
               -if the pronunciation has a helper model, call Training to
               adapt that helper model with all the tokens associated
               with the pronunciation's phonetic or helper model, adding
               the number of such tokens that were successfully used to
               adapt the spelled component as GoodHelperModelTokens~1520
               -if GoodHelperModelTokens and GoodSpelledModelTokens are
               both 0~1522
                    -if pronunciation has a helper model, delete it~1524
                    -call TrainNewModel to build a new helper model for
                    the pronunciation using all of the tokens associated
                    with the pronunciation~1526
               -else, if there is a helper model and GoodHelper-
               ModelTokens is 0~1528
                    -delete the helper model~1530
     -else if Word had no models~1532
          -call TrainNewModel to build a helper model for Word using all
          of the token in the TokenList~1534
     -return~1536
```

FIG. 69

```
-States
     -vocabulary System
          -group System
               -group "Always Active"~1568
                    -"[Command Mode]" /script "CommandMode"~1570
                    -"[Dictate Mode]" /script "DictateMode"~1572
                    -"[Go to Sleep]" /script "GoToSleep"~1574
                    -[Oops] /script "WordHistory 1"~1576
                    -"[What Can I Say]" /script
                    "ShowRecognitionGroups"~1578
               -group "Global Commands"~1580
                    -...
                    -"[Shift Key]" /script "ShiftKey"~1582
                    -"[Alt Key]" /script "AltKey"~1584
                    -"[Control Key]" /script "ControlKey"~1586
                    -...
                    -"a [alpha]"~1588
                    -"b [bravo]"~1588
                    -"c [charlie]"~1588
                    -"d [delta]"~1588
                    -"e [echo]"~1588
                    -"f [foxtrot]"~1588
                    -"g [golf]"~1588
                    -"h [hotel]"~1588
                    -"i [india]"~1588
                    -"j [juliett]"~1588
                    -"k [kilo]"~1588
                    -"l [lima]"~1588
                    -"m [mike]"~1588
                    -"n [november]"~1588
                    -"o [oscar]"~1588
                    -"p [papa]"~1588
                    -"q [quebec]"~1588
                    -"r [romeo]"~1588
                    -"s [sierra]"~1588
                    -"t [tango]"~1588
                    -"u [uniform]"~1588
                    -"v [victor]"~1588
                    -"w [whiskey]"~1588
                    -"x [xray]"~1588
                    -"y [yankee]"~1588
                    -"z [zulu]"~1588
                    -...
                    -"[Spell Mode]" /script "SpellMode"~1590

               -group  "Choice List"~1712
                    -...
                    -"[Choose 1]" /keys {Alt+1}{Enter}
                    -"[Choose 2]" /keys {Alt+2}{Enter}
                    -"[Choose 3]" /keys {Alt+3}{Enter}
                    -"[Choose 4]" /keys {Alt+4}{Enter}
```

FIG. 70

```
                    -"[Choose 5]" /keys {Alt+5}{Enter}
                    -"[Choose 6]" /keys {Alt+6}{Enter}
                    -"[Choose 7]" /keys {Alt+7}{Enter}
                    -"[Choose 8]" /keys {Alt+8}{Enter}
                    -"[Choose 9]" /keys {Alt+9}{Enter}
                    -"[Choose 10]" /keys {Alt+0}{Enter}
                    -...
          -vocabulary Voicebar
               -group Voicebar
                    -...
                    -group "Train Word"~1285
                         -"[Stop Training]" /keys {Alt+s}~1289
                    -...
```

FIG. 70 CONT.

08828130.062697

```
     -AddWordDialog(State)~1316
          -...
          -message loop~1318
               -call MSW GetMessage
               -...
               -if message is "OK"~1320
                    -...
                    -if there is a valid word name string in the Word Name
                    edit box and a valid state selected in the
                    Vocabulary/Group ComboBox~1322
                         -call FindOrMakeMatchingWord for the string to find
                         or make a word ID corresponding to that string~1326
                         -if the word ID is not listed in the selected state,
                         create an entry for it in the selected state~1328
                         -if there is a string in the Resulting Actions edit
                         box, place string in word's ExtraData field in state,
                         preceded by Keystrokes or Script byte, depending upon
                         whether keystroke or Script radio button is
                         selected~1330
                    -remove Add Word dialog box~1332
                    -return~1334
                    -...
               -...
```

FIG. 71

```
-FindOrMakeMatchingWord(String)~1336
     -scan .VOC file for word with a spelling matching String~1338
     -if find one, return with matching word's ID~1340
     -else ~1342
          -create a new word ID in .VOC file, set its spelling equal to
          String, and give it an empty phonetic spelling list~1344
          -if String contains a portion of text inside a top level "[]",
          set String equal to that portion of text~1346
          -strip all punctuation characters besides apostrophes~1348
          -clear IDQueue~1350
          -for each successive word in String~1352
               -scan .VOC file for word with spelling matching the
               successive word~1354
               -if find one, place ID of word in IDQueue~1356
               -else~1358
                    -return with the new word's ID~1360
          -place one empty phonetic spelling in the new word's phonetic
          spelling list~1362
          -for each ID in IDQueue~1364
               -if the ID's word has no phonetic spelling~1366
                    -empty the word's phonetic spelling list~1368
                    -return with the new word's ID~1370
               -for each phonetic spelling of the ID's word~1372
                    -for each prior spelling in the new word's phonetic
                    spelling list~1374
                         -if the total number of spelling's in the
                         phonetic spelling list created in conjunction
                         with the current ID is less than
                         SpellingNumberLimit, create a spelling which
                         concatenates the ID's current phonetic spelling
                         to the end of the prior phonetic spelling,
                         altering phonemes near the boundary of its
                         concatenated spelling if required by
                         coarticulation rules~1376
               -remove the prior phonetic spellings~1378
          -return with new word's ID~1380
```

FIG. 72

```
-FindWordDialog~1550
     -...
     -message loop~1552
          -call MSW GetMessage~1554
          -if message is~1556
               -...
               -"Delete"~1558
                    -if a word has been selected for deletion in
                    conjunction with a given path listed in the
                    Vocabulary/Group ComboBox, delete the selected word
                    from the state indicated in the Vocabulary/Group
                    ComboBox~1560
                    -...
               -...
          -...
```

FIG. 73

```
-ApplicationTracking(HWnd)~1594
    -if HWnd is Null~1596
        -call MSW GetActiveWindow to get the handle of the currently
        active window~1598
        -set HWnd equal to active window handle~1600
    -if HWnd has an entry in HWndToAppTable, return with that entry as
    the SelectedEntry~1602
    -else~1604
        -add a new entry to HWndToAppTable with HWnd, CommandMode as
        its AppMode, an empty AppState, and ShiftKeyOn, ControlKeyOn,
        and AltKeyOn all set to false~1606
        -make the new entry the SelectedEntry~1608
        -call MSW GetWindowWord to get the hinstance of the program
        module running the HWnd's window~1610
        -call MSW GetModuleFileName for that hinstance to get the file
        name of the program which is running HWnd's window~1612
        -compare the file name returned against a list of file names
        associated with stored application states~1614
        -if find a match, set the new entry's AppState equal to the
        state associated with the matching file name~1618
        -else if the file name returned by MSW GetModuleFileName is
        that associated with a MSW file for running MS-DOS applications
        in a window~1620
            -call MSW GetWindowText for HWnd to get the text of its
            window's title bar~1622
            -compare the text returned with a list of text associated
            with application states~1624
            -if find a match, set the new entry's AppState equal to
            the state associated with the matching text~1628
        -if the new entry's AppState is still empty~1630
            -create a new temporary logical state for its
            application~1632
            -set the new entry's AppState equal to the new temporary
            logical state~1634
        -if a call to MSW GetWindow with GW_OWNER for HWnd's window
        indicates the window is a dialog box~1636
            -call MSW GetWindowText for the caption text of the dialog
            box~1638
            -if that text corresponds to the name of a sub-state
            within the AppState of the new entry~1640
                -change the new entry's AppState to that sub-
                state~1642
            -else~1644
                -create a temporary sub-state in the state stored in
                the current entry's AppState~1646
                -place that sub-state in the current entry's
                AppState~1648
    -return with the SelectedEntry~1650
```

FIG. 74

```
-LanguageContextTracking()~1714
     -call MSW GetFocus to get the handle of the window currently having
     the focus~1716
     -use MSW SendMessage to send the focus window the WM_GETDLGCODE
     message to find out if the focus window is a Multi-Line Edit control
     (MLE)~1718
     -if it is an MLE~1720
          -use MSW SendMessage to send EM_GETSEL to the MLE to get the
          character index of the starting position of the current
          selection~1722
          -use MSW SendMessage to send EM_LINEFROMCHAR to the MLE with
          the character index of the start of the current selection to
          get the line number in the MLE of the line on which the current
          selection starts~1724
          -use MSW SendMessage to send EM_GETLINE to the MLE with the
          line number of the current line to get a copy of that line~1726
          -use MSW SendMessage to send EM_LINEINDEX to the MLE with the
          line number of the current line to get the character index of
          start of that line~1728
          -subtract the index of the start of the current line from the
          index of the start of the current selection to determine the
          position in the copy of the current line of the start of the
          current selection~1730
          -starting backward from that position, look in the current line
          for last complete word before the current selection, and if
          that last complete word extends back into the previous line
          look for it in that previous line by using EM_LINEFROMCHAR AND
          EM_GETLINE~1732
          -if there is such a last complete word, set LanguageContext
          equal to it~1734
          -else, set LanguageContext to Null~1736
          -return~1738
     -else if CurrentAppState is associated with an external application
     which has a predefined interface for providing language context~1740
          -send a message to that predefined interface to obtain its
          language context~1742
          -set language context equal to that context~1744
          -return~1746
     -...
     -set LanguageContext to Null~1748
     -return~1750
```

FIG. 75

```
-CommandTracking()~1752
    -clear the CommandPhraseList~1754
    -if a call to MSW GetSystemDebugState returns SDS_MENU, indicating a
    menu is currently active~1756
        -for the menu handle of each entry in MenuStack~1758
            -call GetMenuCommandPhrases~1760
    -else~1762
        -call MSW GetActiveWindow to get the handle of the currently
        active window~1764
        -if a call to MSW GetMenu for the active window returns a menu
        handle, call GetMenuCommandPhrases for the menu~1766
        -if a call to MSW GetSystemMenu returns a menu handle to a copy
        of the system menu, call GetMenuCommandPhrases for the copy of
        the system menu~1768
        -use one or more calls to MSW GetWindow to perform a tree
        search for the handles of all windows, if any, included in
        active window~1770
        -for each window handle obtained~1772
            -if a call to MSW SendMessage sending the window a
            WM_GETDLGCODE message returns an indication the window is
            not a control window, skip to the iteration for the next
            window handle~1774
            -else if a call to IsWindowClickable indicates the window
            is not clickable, skip to the iteration for the next
            window handle~1776
            -else~1778
                -add an empty CommandPhraseEntry in the
                CommandPhraseList~1780
                -call MSW SendMessage to send the window a WM_GETTEXT
                message to get the control's associated text~1782
                -if the value returned in response to the
                WM_GETDLGCODE message indicated the window is a
                static control~1784
                    -if the control's text has an accelerator, save
                    a command to feed the accelerator key to the
                    JournalPlaybackProc in the CommandPhraseEntry's
                    CommandOutput~1788
                    -else~1790
                        -delete the empty CommandPhraseEntry
                        created for this window handle~1792
                        -skip to the iteration for the next window
                        handle~1794
                -call StripControlOrMenuItemName with String equal
                the control's text and TextType equal Control~1796
                -if StripControlOrMenuItemName returns with an empty
                ReturnStringList, delete the current window's
                CommandPhraseEntry and skip to iteration for next
                window~1798
                -else~1800
                    -place the ReturnStringList's first string in
                    the CommandPhraseEntry's CommandPhrase field,
                    enclosed in "[]"~1802
```

FIG. 76

```
                    -if the CommandPhraseEntry's CommandOutput is
                    empty fill it with a "ControlPick[first string]"
                    script command~1804
                    -if the ReturnStringList has a second
                    string~1806
                         -add a copy of the CommandPhraseEntry to
                         the CommandPhraseList and copy the second
                         string enclosed in "[]" into its
                         CommandPhrase field~1808
                         -if the additional CommandPhraseEntry's
                         CommandOutput is empty fill it with a
                         "ControlPick[second string]" script
                         command~1810
     -check to see if there is a tracking state in the tracking state
     cache which includes the exact same collection of command phrases as
     the active window's CommandPhraseList~1812
     -if so~1814
          -make the matching tracking state the CurrentTrackingState~1816
          -set the matching tracking state's LastUsedTime to the current
          time~1818
     -else~1820
          -create a new, empty, tracking state~1822
          -for each CommandPhraseEntry of the CommandPhraseList~1824
               -call FindOrMakeMatchingWord for the CommandPhrase~1826
               -place the word ID, if any, returned by
               FindOrMakeMatchingWord in the new tracking state~1828
               -load the word ID's associated ExtraData field in the new
               tracking state with the value of the CommandPhraseEntry's
               CommandOutput~1830
          -if the tracking state cache has the maximum number of tracking
          states recorded in it, delete from the cache the tracking state
          with the oldest LastUsedTime~1832
          -store the new tracking state in the tracking state cache~1834
          -make the new tracking state the CurrentTrackingState~1836
          -set the new tracking state's LastUsedTime to the current
          time~1838
     -return~1840
```

FIG. 76 CONT.

```
-CommandPhraseList,~1842
     -a list of CommandPhraseEntry structs~1844, each containing
          -CommandPhrase~1846
          -CommandOutput~1848
          -MenuHandle~1850
          -MenuItemPosition~1852
```

FIG. 77

```
-GetMenuCommandPhrases(hmenu)~1860
     -set NumberOK and LastItemWasSeparatorOrNumber to false~1862
     -call MSW GetMenuItemCount to get number of items in the menu for
     which this subroutine was called~1864
     -for each of those items starting with the first~1866
          -call MSW GetMenuItemID to get the menu item's ID~1868
          -if MSW GetMenuItemID returns an indication the menu item is a
          separator, set LastItemWasSeparatorOrNumber to true~1870
          -else~1872
               -create an additional CommandPhraseEntry in the
               CommandPhraseList~1874
               -call MSW GetMenuString to get the menu item's
               spelling~1876
               -if LastItemWasSeparatorOrNumber is true, set NumberOK to
               true~1878
               -else set NumberOK to false~1880
               -call StripControlOrMenuItemName with String equal to the
               menu item's spelling, with TextType equal Menu, and with
               the current value of NumberOK~1882
               -if StripControlOrMenuItemName returns with an empty
               ReturnStringList, delete the CommandPhraseEntry~1884
               -else~1886
                    -place the first string in the ReturnStringList in
                    the CommandPhraseEntry's CommandPhrase enclosed in
                    "[]"~1888
                    -place a "MenuPick[first string]" script command in
                    the CommandPhraseEntry's CommandOutput~1890
                    -place the menu's menu handle in the
                    CommandPhraseEntry's MenuHandle and the menu item's
                    position in the CommandPhraseEntry's
                    MenuItemPosition~1892
                    -if there is a second string in the
                    ReturnStringList~1894
                         -add a copy of the CommandPhraseEntry to the
                         CommandPhraseList~1896
                         -place the second string into the copy's
                         CommandPhrase field enclosed in "[]"~1898
                         -place a "MenuPick[second string]" script
                         command in the copy's CommandOutput~1900
     -return~1902
```

FIG. 78

```
-StripControlOrMenuItemName(String, TextType, NumberOK,
LastItemWasSeparatorOrNumber)~1904
     -if TextType is Menu, if NumberOK is true, and if first character in
     first String is an "&" followed by a numeral and then a space or
     tab~1908
          -set String equal to spelling of the numeral~1910
          -place String in ReturnStringList~1912
          -set LastItemWasSeparatorOrNumber to true~1914
          -return with ReturnStringList~1916
     -set LastItemWasSeparatorOrNumber to false~1917
     -if String contains a top level matching pair of parenthesis~1918
          -place two strings in the ReturnStringList, one corresponding
          to the part of String before the parenthesis, and one
          corresponding to the entire String~1920
     -else place String in the ReturnStringList~1922
     -for each string in the ReturnStringList~1924
          -strip any "&" associated with an accelerator from a
          String~1926
          -strip any leading spaces~1928
          -strip any trailing combination of spaces, periods, colons, and
          exclamation marks~1930
          -strip any character, such as a tab, with a value of 20 Hex or
          less, and any characters following it~1932
          -if the string contains three or more numeric fields separated
          by non-numeric characters remove the string from the
          ReturnStringList~1934
     -return with the ReturnStringList~1938
```

FIG. 79

```
-IsWindowClickable (HWnd)~1940
     -call MSW GetWindowRect to get the screen coordinates of the
     window's bounding rectangle~1942
     -for each of the center point and four corner points of the bounding
     rectangle~1944
          -if a call to MSW WindowFromPoint indicates the window is the
          top window at that point, return with the current point~1946
          -else~1948
               -if using MSW SendMessage to send the WM_NCHITTEST message
               returns HTTRANSPARENT, assume the top window is a group
               box and return with the current point~1950
     -return with an indication that there is no clickable point in the
     window~1952
```

FIG. 80

```
-MenuPick(String)~1954
     -clear the KeystrokeHistoryString~1958
     -if a call to MSW GetSystemDebugState returns SDS_MENU, indicating
     that a menu is currently active~1960
          -for each MenuEntry in MenuStack, starting at the end~1962
               -clear CommandPhraseList~1964
               -call GetMenuCommandPhrases for the MenuEntry's
               MenuHandle~1966
               -for each CommandPhraseEntry placed in the
               CommandPhraseList by GetMenuCommandPhrases~1968
                    -if the spelling within the "[]" of its CommandPhrase
                    matches String~1970
                         -add to the KeystrokeHistoryString the arrow
                         keystrokes necessary to move from the position
                         of the MenuEntry's MenuItemID to that associated
                         with the CommandPhraseEntry's
                         MenuItemPosition~1972
                         -add "enter" to the KeystrokeHistoryString~1974
                         -use the JournalPlaybackProc to playback the
                         KeystrokeHistoryString~1976
                         -return~1978
               -add an "escape" key to the KeystrokeHistoryString~1980
               -delete the last MenuEntry from the end of the
               MenuStack~1982
     -else~1984
          -call MSW GetActiveWindow, GetMenu, and GetSystemMenu to get
          the active window's main menu and its system menu~1986
          -clear the CommandPhraseList~1988
          -for the active window's menu call GetMenuCommandPhrases~2000
          -for the active window's system menu call
          GetMenuCommandPhrases~2002
          -for each CommandPhraseEntry in the CommandPhraseList~2004
               -if the spelling within "[]" of its CommandPhrase matches
               String~2006
                    -if the CommandPhraseEntry's MenuHandle is that of
                    active window's main menu, add to the
                    KeystrokeHistoryString an "Alt" followed by the arrow
                    keystrokes necessary to go from first item in the
                    menu to the CommandPhraseEntry's MenuItemPosition,
                    followed by an "Enter"~2008
                    -else if its menu handle is that of the active
                    window's system menu, add to the
                    KeystrokeHistoryString an "Alt-Spacebar" followed by
                    the arrow keystrokes necessary to go from the first
                    item in the system menu to the item represented by
                    the MenuItemID of the matching CommandPhraseEntry,
                    followed by an "Enter"~2010
                    -use the JournalPlaybackProc to play keystrokes back
                    to active application~2012
                    -return~2014
     -display an error message~2016
     -return~2018
```

FIG. 81

```
-ControlPick(String)~1956
     -call MSW GetActiveWindow to get the handle of the currently active
     window~2020
     -use one or more calls to MSW GetWindow to perform a tree search for
     the handles of all child windows, if any, included in the active
     window~2022
     -for each child window handle obtained~2024
          -if using MSW SendMessage to send the child window the
          WM_GETDLGCODE message returns an indication the child window is
          not a non-static control, skip to the iteration for the next
          child window~2026
          -call MSW SendMessage to send the child window a WM_GETTEXT
          message to get the control window's associated text~2028
          -call StripControlOrMenuItemName with window's text as String
          and with TextType equal to Control~2030
          -if any string in the ReturnStringList returned by
          StripControlOrMenuItemName matches the String with which
          ControlPick was called~2032
               -if a call to IsWindowClickable for the window returns a
               clickable point, uses the JournalPlaybackProc to send the
               window the WM_LBUTTONDOWN and then the WM_LBUTTONUP
               messages at that point~2034
               -return~2036
     -if no control window with text matching ControlPick's String is
     found, display an error message.~2038
     -return~2040
```

FIG. 82

```
-PropertiesTabOfAdvancedModifyWordDialog(Word, State)~2054
     -...
     -message loop~2056
          -call MSW GetMessage~2058
          -if message is~2060
               -...
               -OK~2062
                    -if Forget Training button is pressed, remove word's
                    helper model from .USR file~2064
                    -...
               -...
          -...
     -...
```

FIG. 85

```
-if Forget Training button is pressed, remove word's
helper model from .USR file and reset the PIC and PEL
counts on each of the word's PIC's and PEL's~2064A
```

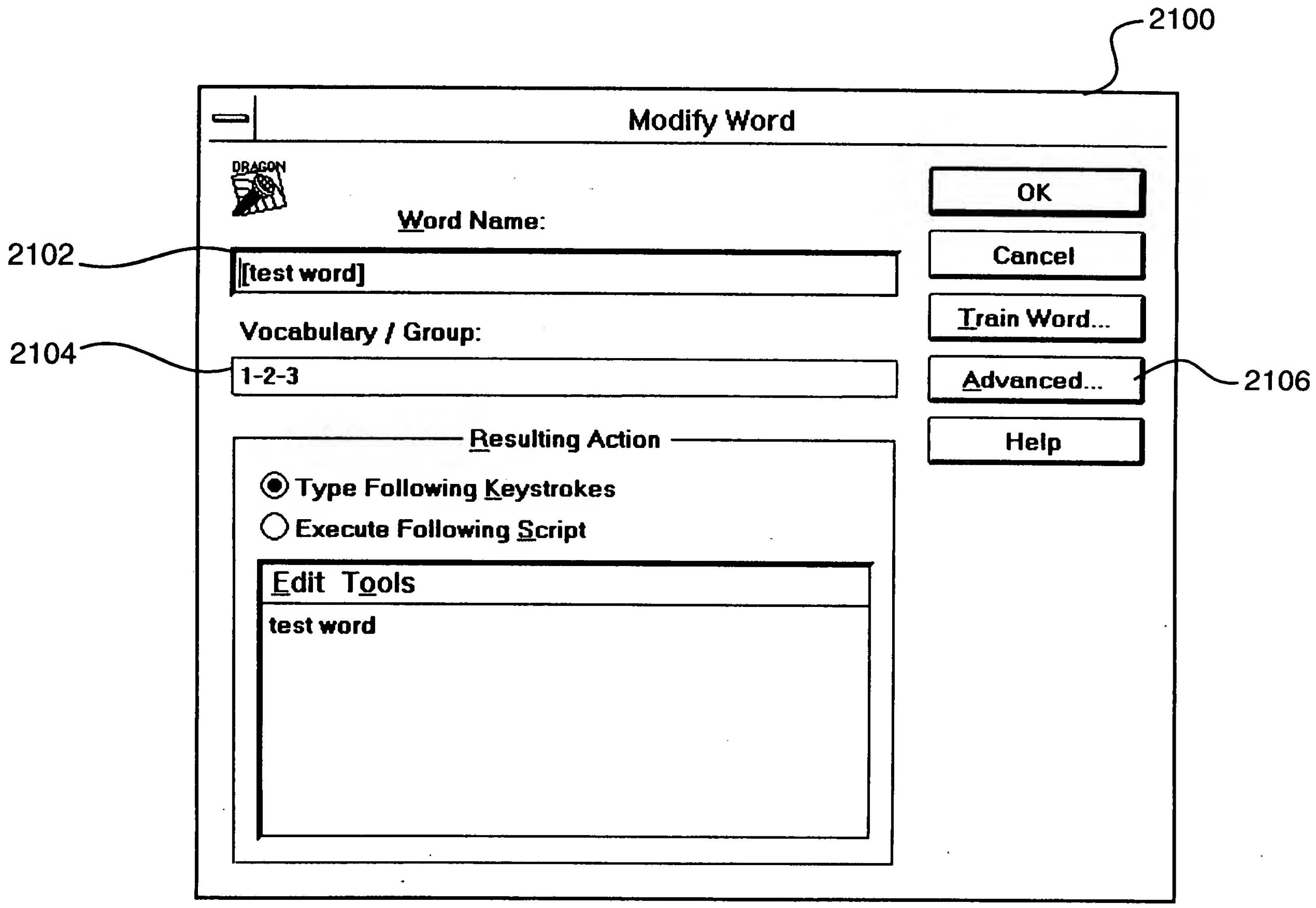
FIG. 85A

```
-SlowDemon()~2074
          -...
          -if HandsFree is true, RecognizerOn is false, MicOffConfirmed
          is false, and if (either there are no MacroInstances or there
          is at least one MacroInstance waiting for user input), call
          MicrophoneWarning~2076
          -...
```

FIG. 87

```
-MicrophoneWarning()~2078
     -set CurrentMode to CommandMode~2080
     -set RecognizerOn to true~2082
     -call MSW MessageBox to display, get input from, and remove
     Microphone Warning message box~2084
     -if MSW MessageBox returns with~2086
          -Yes~2088
               -set RecognizerOn to false~2088
               -set MicOffConfirmed to true~2090
     -return~2092
```

FIG. 88



Modify Word

Word Name: [test word]

Vocabulary / Group: 1-2-3

Resulting Action

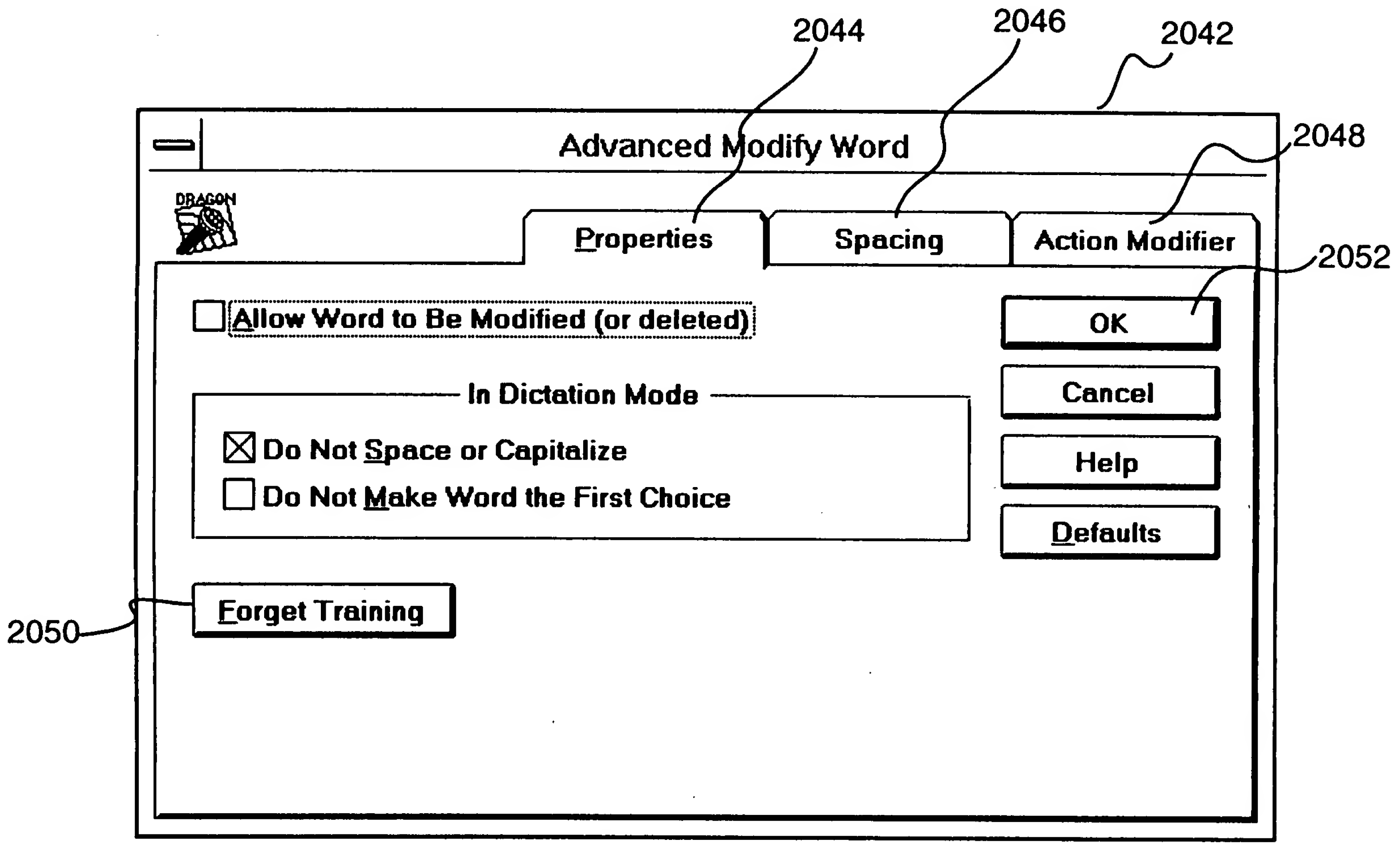
[✓ Type Following Keystrokes]
Execute Following Script

Edit Tools

test word

Figure 83

APPROVED	O.G. FIG.	
BY	CLASS	SUBCLASS
DRAFTSMAN		



Advanced Modify Word

☐ Allow Word to Be Modified (or deleted)

In Dictation Mode

☒ Do Not Space or Capitalize

☐ Do Not Make Word the First Choice

Figure 84

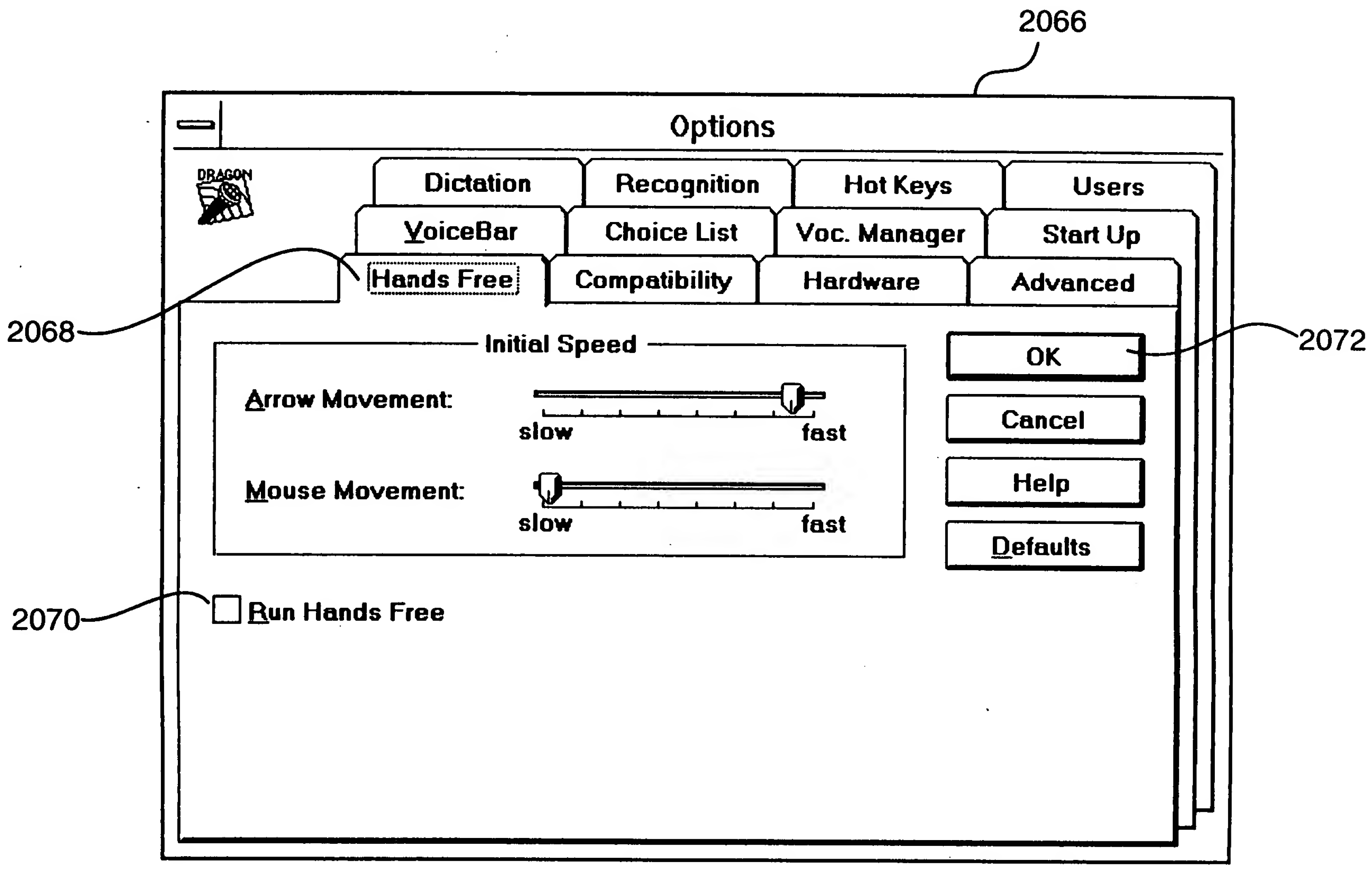
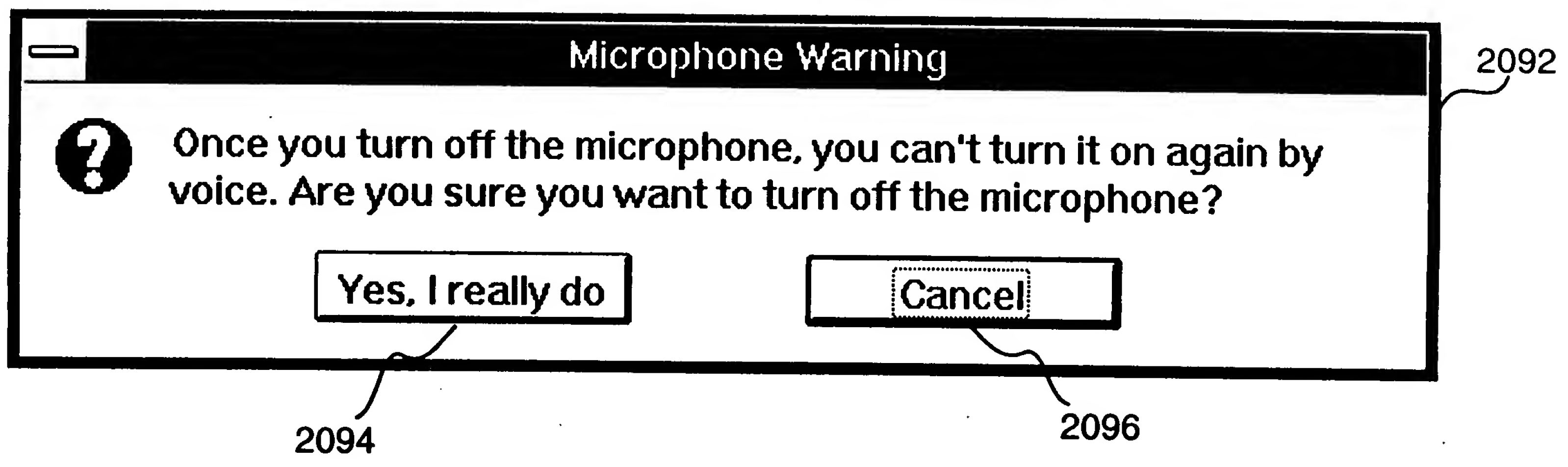

2066
Options
Dictation
Recognition
Hot Keys
Users
VoiceBar
Choice List
Voc. Manager
Start Up
Hands Free
Compatibility
Hardware
Advanced
2068
Initial Speed
Arrow Movement:
slow
fast
Mouse Movement:
slow
fast
OK
Cancel
Help
Defaults
2072
2070
Run Hands Free

Figure 86

08882813.062697



Once you turn off the microphone, you can't turn it on again by voice. Are you sure you want to turn off the microphone?

Figure 89